60449900

CONTROL DATA
CORPORATION

UPDATE 1
REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1
SCOPE 2

UPDATE DIRECTIVES INDEX

Directive	Parameters	Abbreviation	Page
*ABBREV		none	3-12
*ADDFILE	lfn,c	*AF	3-5
*ADDFILE	lfn,deck	*AF	3-5
*BEFORE	c	*B	3-5
*CALL	deck	*CA	3-10
*CHANGE	oldid,newid, . . . ,oldid,newid	*CH	3-5
*COMDECK	deck	*CD	3-4
*COMDECK	deck,NOPROP	*CD	3-4
*COMPILE	deck1.deck2	*C	3-10
*COMPILE	deck1,deck2, . . . ,deckn	*C	3-10
*COPY	deck,c	*CY	3-6
*COPY	deck,c1,c2	*CY	3-6
*COPY	deck,c1,c2,lfn	*CY	3-6
*CWEOR	level	*CW	3-10
*DECK	deck	*DK	3-4
*DECLARE	deck	*DC	3-13
*DEFINE	name1,name2, . . . ,namen	*DF	3-13
*DELETE	c1,c2	*D	3-6
*DELETE	c	*D	3-6
*DO	ident1,ident2, . . . , identn	none	3-11
*DONT	ident1,ident2, . . . ,identn	*DT	3-11
*END		none	3-13
*ENDIF		*EI	3-11
*ENDTEXT		*ET	3-12
*IDENT	idname,p1,p2, . . . ,pn	*ID	3-7
*IF	type,name,num	none	3-11
*IF	-type,name,num	none	3-11
*INSERT	c	*I	3-7
*LIMIT	n	*LT	3-13
*LIST		*L	3-12
*MOVE	deck1,deck2	*M	3-7
*NOABBREV		*NA	3-13
*NOLIST		*NL	3-13
*PULLMOD	ident1,ident1, . . . ,identn	*PM	3-14
*PURDECK	deck1,deck2, . . . ,deckn	*PD	3-7
*PURDECK	deck1.deck2	*PD	3-7
*PURGE	ident1,ident, . . . ,identn	*P	3-8
*PURGE	ident1.ident2	*P	3-8
*PURGE	ident,*	*P	3-8
*READ	lfn	*RD	3-12
*RESTORE	c	*R	3-8
*RESTORE	c1,c2	*R	3-8
*REWIND	lfn	*RW	3-12
*SELPURGE	deck.ident1,deck2.ident2, . . . ,deckn.identn	*SP	3-8
*SELYANK	deck1.ident1,deck2.ident2, . . . ,deckn.identn	*SY	3-9
*SEQUENCE	deck1,deck2, . . . ,deckn	*S	3-9
*SEQUENCE	deck1.deck2	*S	3-9
*SKIP	lfn,n	*SK	3-12
*TEXT		*T	3-13
*WEOR	level	*W	3-12
*YANK	ident1,ident2, . . . ,identn	*Y	3-9
*YANK	ident1.ident2	*Y	3-9
*YANKDECK	deck1.deck2, ...deckn	*YD	3-10
*/	comments	none	3-14

REVISION RECORD

REVISION	DESCRIPTION
A (12-15-75)	Original printing. This manual is a successor to publication number 60342500 for users of NOS 1.0, NOS/BE 1.0, and SCOPE 2.1 operating systems.
B (3-31-78)	This revision reflects version 1.3 of the Update utility at PSR level 472. Update has been modified to allow up to seven secondary old program libraries to be specified. The entire manual has been reprinted.
Publication No. 60449900	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning
this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the
back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Cover	–
Inside Cover	–
Title Page	–
ii	B
iii/iv	B
v/vi	B
vii	B
viii	B
1-1 thru 1-3	B
2-1 thru 2-4	B
3-1 thru 3-14	B
4-1 thru 4-5	B
5-1 thru 5-6	B
A-1	B
A-2	B
B-1 thru B-7	B
C-1	B
D-1 thru D-8	B
E-1	B
Index-1	B
Index-2	B
Comment Sheet	B
Return Env	–
Inside Cover	–
Cover	–

Page	Revision

Page	Revision

PREFACE

This manual describes the Update utility for maintaining and updating decks in compressed symbolic format on mass storage. As described in this publication, Update 1.3 operates under the control of the following operating systems:

> NOS 1 for the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174, and 175; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems
>
> NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems
>
> SCOPE 2 for the Control Data CYBER 170 Model 176; CYBER Model 76; and 7600 Computer Systems

The user is assumed to be familiar with the operating system and computer system in use.

Related material is contained in the following publications:

Publication	Publication Number
NOS 1 Reference Manual, volume 1	60435400
NOS 1 Reference Manual, volume 2	60445300
NOS/BE 1 Reference Manual	60493800
SCOPE 2 Reference Manual	60342600

CDC manuals can be ordered from Control Data Literature and Distribution Services, 8001 East Bloomington Freeway, Minneapolis, Minnesota, 55420.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION 1

Update is a utility for maintaining and manipulating a mass storage file containing images of coded punched cards or their equivalent. Once card images have been made a part of an Update program library, any physical punched cards can be dispensed with.

A file of card images to be manipulated by Update must be in a special format known as a program library. Three types of Update runs generate or manipulate a program library:

A creation run generates a program library from the input stream text.

A correction run manipulates the contents of an existing program library.

A copy run changes the format of a program library from random to sequential or from sequential to random.

As each card is written to the program library, Update assigns it a unique card identifier.

Groups of card images within the program library are known as decks. Each program library must have at least one deck, with the maximum number of decks being 262,143. Deck grouping is significant in terms of extracting card images from the program library in a format suitable for use by a compiler, assembler, or print routine. While an individual card can be referenced for purposes such as deletion of that card or insertion after that card, the smallest unit that can be extracted from the program library is the deck.

Typically, use of Update involves maintenance of a group of compiler or assembly language routines. For convenience, the programmer often specifies each routine or group of related routines as an individual deck. One routine can then be changed or extracted without affecting other routines in the program library. Because each card image in a deck has its own identifier (consisting of the deck name) and an Update-supplied sequence number, it can be referenced individually in order to correct or change a routine. Then, the deck containing the modified routine can be extracted from the program library and used as if it had been entered into the system as a punched deck.

A deck can be composed of punched cards or images of punched cards. Update makes no assumptions about contents. While programs are customary contents, Update is equally applicable to a set of data cards or any other text.

The programmer controls Update operations through the following two mechanisms:

The UPDATE control statement parameters specify the general operations to be performed. The parameters control the files to be manipulated and influence the type of operations performed.

The input stream card images specify the detailed operations to be performed and specify the card images to be made a part of the program library. The instructions for Update operation are called directives; the card images for the program library are called text. The input stream can be either part of the job deck containing the UPDATE control statement or a separate file.

Files used or generated by Update have generic names that are related to their default logical file names. The following names are used in the remainder of this manual in describing Update operations:

Input file – the user-supplied file or part of the job deck that contains the input stream of Update directives and text.

Output file – the file generated by Update that contains the status information produced during Update execution. It is in a format suitable for printing.

Program library – the file generated by an Update creation run that contains the decks of card images. At the time the file is created, it is known as the new program library. At the time the file is corrected, it is known as the old program library. A correction run can optionally create a separate new program library that permanently incorporates the changes made during the correction run. Card images in the program library are in a format that can be manipulated by Update, but that is meaningless for all other purposes.

Compile file – the file generated by Update that contains card images restored to a format that is acceptable to a compiler or assembler. Decks written to the compile file during any given run are controlled by the Update mode selected, by control statement parameters, and directives in the input stream.

Source file – the file generated by Update that contains card images of an input stream that would allow regeneration of the program library.

Section 2 contains a detailed discussion of the files used or generated by Update.

The directives for Update are interspersed with text in the input stream. They are distinguished by the presence of a control character contiguous with a directive keyword. More than 40 directives exist. The directives can be grouped into those that

Define decks.

Control compile file contents.

Manipulate primary or secondary input streams.

Control overall handling of input files.

Modify program library contents.

Section 3 contains a detailed discussion of Update directives.

CREATION RUN

A creation run constructs a program library. It is the original transfer of punched cards or card images into Update format.

A creation run exists when the first card read from the input stream is a DECK or COMDECK directive. A creation run also exists when one or more of the following directives precedes the first DECK or COMDECK directive:

ABBREV
DECLARE
LIMIT
LIST
NOABBREV
NOLIST
READ
REWIND
SKIP
/ (comment)

The presence of any other directive before the first DECK or COMDECK directive causes Update to consider the run to be a correction run.

In addition to the preceding directives, the following are the only Update directives that can be used during a creation run:

CALL
CWEOR
END
ENDIF
ENDTEXT
IF
TEXT
WEOR

Each DECK or COMDECK directive defines a deck to be inserted into the program library under construction. All text and directives following a DECK or COMDECK directive, until the next DECK or COMDECK directive, are considered to be part of the deck. Each card image receives the deck name and a unique sequence number so that the images can be referenced individually. The DECK or COMDECK directive defining the deck itself is assigned the sequence number one.

Update decks can be one of two types: a regular deck declared with a DECK directive, or a common deck declared with a COMDECK directive. DECK and COMDECK differ in that a common deck can be called by name so that it is inserted into the text of another deck when the compile file is being generated. One copy of the common deck exists on storage, but multiple copies can be part of a compile file.

When the library is created, Update generates a deck named YANK$$$ as the first deck on the library. The YANK$$$ deck contains all the YANK, SELYANK, YANKDECK, and DEFINE directives that are encountered during Update runs. (The YANK$$$ deck is described further in appendix D, File Format and Structure.) Update also generates a deck list and directory during a creation run. The deck list contains the names of all decks in the library and the location of the first word for each deck. The directory contains one entry for each DECK, COMDECK, and IDENT directive that is used for the library.

CORRECTION RUN

A correction run, which is the most common use of Update, introduces changes into the existing program library. These changes exist only for the duration of the run unless a new program library is generated. Update recognizes a correction run when it encounters a directive other than one of the following prior to encountering DECK or COMDECK:

ABBREV
DECLARE
LIMIT
LIST
NOABBREV
NOLIST
READ
REWIND
SKIP
/

A correction run consists of a read-input-stream phase and a correction phase. During the first phase, Update reads directives and text, adds any new decks, and constructs a table of requested correction operations. During the second phase, Update performs the requested modifications on a deck-by-deck basis.

The corrections to the library, that is, the newly inserted cards, replaced cards, and deleted cards make up correction sets. The IDENT directive assigns a unique identifier to each card image inserted by the correction directives. Each inserted card image is assigned a sequence number beginning with one for each IDENT name. All card images having the same correction set identifier comprise a correction set.

Update permits a user to remove (yank) the effects of a correction set or deck and later restore the correction set or deck. This feature is convenient for testing new code. Requests for yanking are maintained in the YANK$$$ deck. Before obeying a correction, Update checks the correction identifier against the YANK$$$ deck to see if the correction has already been yanked. If it has been yanked, an informative message is issued and processing continues. This effect on the YANK$$$ deck can be selectively controlled through DO and DONT directives within the decks.

The image of a card, even though deleted through DELETE or yanking, is maintained permanently on the program library with its current status (active or inactive) and a chronological history of modifications to its status. The images contain information known as correction history bytes. The history bytes, which are generated by Update, contain the history and status of the card and is the means by which Update can reverse status. Deletion of a card, for example, is accomplished by the addition of a correction history byte to the card image rather than a physical deletion of the image. Consequently, the card can be reactivated at some later time.

Update also allows a complete and irreversible purging of correction sets and decks. When a correction set or deck is purged, it is physically removed from the library.

COPY RUN

A copy run changes the old program library format from sequential to random or from random to sequential. Update recognizes a copy run when either the A or B parameter is specified on the UPDATE control statement. Since Update does not read the input file on a copy run, no other operations are performed.

DECK LIST AND DIRECTORY ORDER

Update maintains a deck list and directory for its internal use. They are only significant to the user when ranges of decks or correction sets are specified on Update directives. The output file lists the order in which the deck names and correction set identifiers appear.

The deck list contains a list of all decks in the program library. The original entries of the deck list correspond to the order in which decks were written during the creation run. Subsequent entries are added to the end of the list as they are introduced in the program library. Therefore, deck list order might not reflect actual deck order in the program library, since the user determines deck location within the program library through directives.

The location of an entry in the deck list is significant in terms of parameters for PURDECK, SEQUENCE, and COMPILE directives in which a range of decks can be referenced. The order of names in a range reference must be the same as the order in the deck list. The decks named and all the decks between are then processed in accordance with the directive. An error exists if they are in reverse order.

Similarly, as each deck and correction set is introduced into the program library, Update creates an entry in an internal directory in chronological sequence. The location of an entry in the directory is significant in terms of parameters for PURGE and YANK directives in which a range of correction sets can be referenced. The order of reference must be the same as the order of the directory. The identified correction sets and all the sets between are processed in accordance with the directive. An error exists when a correction set range is not referenced in the order the sets were introduced into the library.

UPDATE MODE

The content of any compile file, source file, or new program library produced during a correction run is affected by the Update mode. (Table 2-2 summarizes the effect of mode upon file content.) The mode of an Update run is determined by a combination of the omission or specification of the F and Q parameters on the Update control statement as summarized in table 1-1.

Mode chosen depends on how extensively the user wishes to modify the program library and its size. If the library contains many decks and the user wishes to modify only a few, quick mode would be used. If there are many decks and the user wants all decks to be processed, full mode would be used. Normal selective mode would be used when only those decks modified or specified are wanted in the compile file.

TABLE 1-1. UPDATE MODE

Parameter	Mode
F specified	Full mode in which all decks are on the old program library are processed.
Q specified	Quick mode in which only decks specified on COMPILE directives and decks added through ADDFILE directives are processed.
Both F and Q specified	Quick mode.
Both F and Q omitted	Normal selective mode in which the only decks processed are those modified or those specified on COMPILE directives.

UPDATE FILES 2

During its execution, Update manipulates as many as eight files that can be referenced by the user. The files involved with any given run depend on

Parameters selected by the UPDATE control statement

Whether the run is a creation run, correction run, or copy run

The files that Update generates or uses are described below. Each of these files has a default name, but other names can be specified through the appropriate parameters on the UPDATE control statement.

File characteristics are summarized in table 2-1.

Whether or not a file is optional, used, or not applicable on an Update run depends on the type of run.

Creation run – the user must supply the input file. Update generates the new program library, compile file, and output file by default. The generation of a source file is optional. No other files are applicable on a creation run.

Correction run – the user must supply the input file, the old program library, and the merge file (if a merge is to take place). Update generates, by default, the output and compile files. The creation of a new program library, source file, and pullmod file is optional on a correction run.

Copy run – the user must supply the old program library. Update generates, by default, the new program library and the output file. No other files are applicable on a copy run.

The contents of any compile file, source file, or new program library produced during a run are affected by the Update mode and the file format of the old program library. The contents of these files are summarized in table 2-2.

INPUT FILE

The input file contains the input stream; it must contain coded cards or their equivalent. The input stream consists of directives that direct Update processing and text to be added to the program library. The directives allowed in the input stream are determined by the type of Update run.

Update initially reads the input stream from the primary input file specified by the I parameter of the UPDATE control statement; default file name is INPUT. Update stops reading directives and text when it encounters a 7/8/9 card or its equivalent, or end-of-information (EOI).

If Update encounters a READ or ADDFILE directive in the input stream, it stops reading from the primary input file and starts reading from the file specified on the directive. Update reads one system-logical-record from the secondary input file; reading from the primary input file then resumes.

TABLE 2-1. FILE SUMMARY

File	Default Name	Contents	Type	Default Position
Input	INPUT	The input stream	Coded	Remains at the end of the record terminating Update directives. If Update aborts, location of input file is unpredictable.
New program library	NEWPL	Updated library	Binary	Rewound before and after run.
Old program library	OLDPL	Library to be updated	Binary	Rewound before and after run.
Secondary old program library	None	Library from which common decks can be called.	Binary	Rewinding not necessary because file must be random.
Compile	COMPILE	Card images for assembly or compilation	Coded	Rewound before and after run.
Output	OUTPUT	Information for use by the programmer	Coded	Remains in current position. File is not rewound.
Source	SOURCE	Consists of card images that would allow regeneration of a new program library	Coded	Rewound before and after run.
Merge	MERGE	Second library to be merged into new program library	Binary	Rewound before and after run.
Pullmod	Source file	Re-created correction sets	Coded	Rewound before and after run.

TABLE 2-2. FILE CONTENTS AND UPDATE MODE

File	Normal Mode Contents	Full Mode Contents	Quick Mode Contents (sequential OLDPL)	Quick Mode Contents (Random OLDPL)
New program library	Regular decks and common decks after corrections are made	Regular decks and common decks after corrections are made	All decks specified on COMPILE directives, any common decks they call, and any common decks encountered prior to all decks of COMPILE	Decks specified on COMPILE directives and any common decks they call
Compile File	Decks corrected or on COMPILE directives and decks calling a corrected common deck (unless the calling deck precedes the common deck or NOPROP is specified on COMDECK)	Active decks on old program library	Decks on COMPILE directives and decks added via ADDFILE plus called common decks	Decks on COMPILE directives and decks added via ADDFILE plus called common decks
Source File	Active cards and deck required to recreate the library	Active cards and decks required to recreate the library	Active cards from decks specified on COMPILE directives, any common decks they call, and any common decks encountered prior to all decks on COMPILE	Active cards from decks specified on COMPILE directives and any common decks they call

PROGRAM LIBRARY FILES

A program library is created during an Update run and can be manipulated in later runs. The library consists of a file of card images and internal information in a special format that can be processed only by Update. The card images are grouped into decks. Each card image is represented in a blank-compressed format that adds a card identifier. The format also includes history and status information that is known as correction history bytes.

The program library also contains a deck list and a directory. The deck list contains the names of all decks in the library. In addition to deck names, the directory also contains the names of all correction sets. Unless changed by the E parameter of the UPDATE control statement, the names in both the deck list and the directory exist in the order they were introduced.

Update can create and maintain program library files in two distinct formats: random and sequential. (These formats are described in detail in appendix D.) A random program library can be processed substantially faster than a sequential program library.

NEW PROGRAM LIBRARY

A new program library is initially generated on a creation run. It contains directives and text in an updatable format. File content is determined by the file format of the old program library and Update mode as shown in table 2-2. New program library name is specified by the N parameter of the UPDATE control statement; default file name is NEWPL.

For subsequent correction runs, the previously generated new program library is identified as the old program library. A new program library that incorporates the changes made during the correction run is generated if requested.

A new program library can be in random or sequential format. In the absence of the W parameter on the UPDATE control statement, the format is determined by file residence and record type as shown in table 2-3.

TABLE 2-3. NEW PROGRAM LIBRARY FORMAT

Format	NOS and NOS/BE	SCOPE 2
Random	File is on mass storage and W is not selected	File is on mass storage, record type is W unblocked, and W is not selected
Sequential	File is on magnetic tape or W is selected	File is staged or on-line tape; or is on mass storage as record type S or record type W blocked; or W is selected, or R specifies no rewind

A random new program library cannot be written to an existing permanent file. A new program library can be made into a permanent file by saving it after the Update run has been completed. In contrast, a sequential new program library can be added to an existing permanent file by writing the program library at end-of-information. (See the operating system reference manual for details.)

OLD PROGRAM LIBRARY

The old program library is the file that was generated as a new program library in a previous run. It contains a record of changes made since the program library was created. Old program library name is specified by the P parameter of the UPDATE control statement; default file name is OLDPL.

An old program library is required for a correction run since it is the program library to be updated. On a copy run, the old program library is not modified, but is copied to a sequential or random new program library. If an old program library is specified on a creation run, it is ignored.

In addition to the old program library to be updated, up to seven additional (secondary) old program libraries can be specified by the P parameter of the UPDATE control statement. Decks on the old program library can call common decks from the old program library or from any of the other program libraries. No Update directive other than CALL can be used to reference common decks on secondary old program libraries. Common decks on secondary old program libraries can call common decks which reside on any of the old program libraries. Program libraries are searched in the order specified to find the called common decks. The called common decks that reside on the secondary old program libraries are not added to a new program library.

The secondary old program libraries must be random, have a unique name, and have the same master control character as the old program library. If these conditions are not met, a message is issued.

When creating a new program library on a creation run that contains calls to common decks that reside on secondary old program libraries, C=0 must be specified on the UPDATE control statement.

COMPILE FILE

The compile file contains copies of decks in the program library restored to a format that can be processed by a compiler or assembler. The decks written to the file are determined by Update mode and the file format of the old program library as shown in table 2-2. Through the control statement parameters D and 8, the user can specify whether the text on the file is to have Update card identifiers on each line of text.

Compile file name is specified by the C or K parameter of the UPDATE control statement; default file name is COMPILE. If the K parameter is specified, then decks are written to the compile file in the order they appear on COMPILE directives. (Any decks not specified on COMPILE directives follow those specified.) If the C parameter is specified, then decks are written on the compile file in the order they appear in the deck list.

The user has control over the decks written to the compile file through the compile file directives. Common decks can be called conditionally or unconditionally according to compile file directives embedded in the program library decks. Additional control of compile file format is afforded the user through directives that cause a system-logical-record of the specified level to be written at the end of decks. The compile file directives can be in the original decks or can be inserted into the program library decks during correction runs. These directives are interpreted when the compile file is written; they are not written on the compile file.

LISTABLE OUTPUT FILE

The listable output file is the print file containing information for use by the programmer. Content of the file is controlled by the L parameter of the UPDATE control statement with options that can select a listing of directives processed, errors, comments, and a list of card images in the program library. The locations of all CWEOR, WEOR, and CALL directives are listed if a compile file is written. If L=0, all listable output is suppressed. Output file name is specified by the O parameter of the UPDATE control statement; default file name is OUTPUT.

In quick mode only, Update produces an ordered printout of the deck list of the program library under the heading DECK LIST AS READ FROM OLDPL PLUS ADDED NEW DECKS. A quick mode dummy Update run (no decks added) produces a deck listing of the old program library.

SOURCE FILE

The source file is an optional file generated during a correction or creation run. The source file consists of the card images of an input stream that would allow generation of a new program library with only currently active card images in resequenced format during a subsequent creation run. Only active DECK, COMDECK, WEOR, CWEOR, CALL, TEXT, IF, ENDIF, and ENDTEXT directives, in addition to all active text, are part of the source file. The card images in the source file do not contain card identifiers.

Source file name is specified by the S parameter of the UPDATE control statement; default file name is SOURCE. The content of the file is determined by the T parameter of the UPDATE control statement and by Update mode and the file format of the old program library as shown in table 2-2. The user is responsible for routing the source file to a punch or other output device.

MERGE FILE

The merge file contains a program library to be merged with the old program library into a new program library. Update adds the deck list and directory from the merge file to the deck list and directory on the old program library. Any names on the merge file that duplicate names on the old program library are modified to make them unique as follows:

> The last character of the name is changed by adding 01 (modulo 55_8) until all valid characters have been tried.
>
> A character is appended to the name and the first step is repeated. Characters are appended until the name reaches nine characters.

If no unique name can be generated by this method, the Update run is abnormally terminated. Directives that reference these changed names are modified to agree with the new name. All names that required modification are listed in the output file.

Merge file name is specified by the M parameter of the UPDATE control statement; default name is MERGE. All Update functions that are valid in a correction run are valid with the merge parameter. Care should be exercised when including modifications in a merge run. Update might change a name to which correction cards have been applied. In this case, corrections can refer to the wrong deck or correction set.

Decks from the merge file are added to the new program library after all decks from the old program library are added. This sequence of decks in the new program library can be altered by the MOVE directive if desired.

PULLMOD FILE

The pullmod file contains directives and text of re-created correction sets specified on PULLMOD directives. These re-created correction sets produce the same results as the original sets. The file has the same format as an input file. This feature permits a user to take an earlier version of the library and apply selected correction sets.

File name is specified by the G parameter of the UPDATE control statement. If no file is specified, pulled modifications are written to the source file specified by the S or T parameter; if no source file is specified, the re-created correction sets are written to a file named SOURCE.

SCRATCH FILES

Update uses six scratch files. These files are not available to the user. They are summarized in table 2-4.

TABLE 2-4. UPDATE SCRATCH FILES

File Name	Function
UPDTSCR	Used to make copy of decks to be written later to compile file.
UPDTCDK	Used to hold common decks for later expansion of CALL directives.
UPDTTPL	Used as a temporary program library.
UPDTEXT	Used to copy card images to be inserted ir correction run.
UPDTAUD	Used to hold temporary audit information.
UPDTPMD	Used to collect card images in response to PULLMOD directives.

DIRECTIVES 3

Directives allow the user to create program libraries. Directives also extensively control and direct the correction and modification process. Directives perform the following operations:

- Identify decks.
- Control compile file contents.
- Manipulate primary or secondary input streams.
- Control overall handling of the input file.
- Modify program library contents.

Each directive is summarized in table 3-1.

DIRECTIVE FORMAT

The general format of Update directives is shown in figure 3-1. A directive must begin with the master control character in column one. Comments can be placed after the last parameter of the directive. The comment and final parameter must be separated by one or more blanks. Most directives have both a full keyword and an abbreviated keyword as shown in table 3-1; when the NOABBREV directive is in effect, Update does not recognize the abbreviated forms of directive names. Any card in the input stream that cannot be recognized as a directive is assumed to be text.

*keyword p-list

*	Master control character which distinguishes a directive from a text card. Must appear in column 1. This character can be changed through the * parameter of the UPDATE control statement.
keyword	Name of one of the Update directives or an abbreviation for a directive. No blanks can occur between the master control character and the keyword; a comma or blank terminates the keyword.
p-list	Parameters identifying decks, cards, or files. Some directives have no parameters. Multiple blanks can appear between the keyword and parameters. Parameters in the list are separated by commas; embedded blanks cannot appear in the list. A blank terminates the p-list. Notice that several parameters contain a period as part of a single parameter.

Figure 3-1. General Update Directive Format

The master control character is recorded in the program library. For a correction run, the master control character should match the character used when the program library was created. If the characters do not match, Update uses the character specified in the program library.

Since Update scans all 80 columns when interpreting directives, comments or sequencing information from a previous run can be interpreted as the parameter list. Update interprets comments or sequencing information as the parameter list when a list is not specified on WEOR, CWEOR, DECLARE, or ADDFILE directives. To avoid this problem, a null parameter list should be specified on these directives in the following manner:

```
*WEOR, ,        *DECLARE, ,
*CWEOR, ,       *ADDFILE, , ,
```

Specifying a null parameter field ensures that Update will use the default values as parameters rather than using the comments or sequencing information. Errors will occur if Update tries to use the comment or sequencing information as the directive parameter list.

CARD IDENTIFIERS

Each card image in a program library is uniquely identified by an identifier and a sequence number. The identifier is the name of the deck or correction set from which the card image originated; Update supplies the sequence number. Card identifiers assigned by Update are usually permanent; they can be changed only through the use of the SEQUENCE and CHANGE directives.

Update recognizes one full form and two short forms of card identifiers. The full form card identifiers are shown in figure 3-2. The two short forms of card identifiers, which can be used on BEFORE, INSERT, DELETE, RESTORE, and COPY directives, are expanded as shown in figure 3-3.

ident.seqnum

ident.	1 through 9 character name of a correction set or deck. A period terminates the identifier.
seqnum	Decimal ordinal (1 through 131071) representing the sequence number of the card within the correction set or deck. Any character other than 0 through 9 terminates the sequence number.

Figure 3-2. Full Form of Card Identification

seqnum	Expands to idname.seqnum where idname is a correction set identifier, whether or not it is also a deck name.
.seqnum	Expands to dname.seqnum where dname is a deck name.

Figure 3-3. Expansion of Short Forms of Card Identification

TABLE 3-1. SUMMARY OF UPDATE DIRECTIVES

Directive Keyword Abbreviation	Directive Format	Use
none	*ABBREV	Resume checking for abbreviated directives.
*AF	*ADDFILE lfn,name	Read creation directives and text from named file and insert after specified deck or card.
*B	*BEFORE c	Write subsequent text cards before card identified.
*CA	*CALL deck	Write common deck to compile file.
*CH	*CHANGE oldid,newid, . . . ,oldid,newid	Change correction set identifier.
*CD	*COMDECK deck,NOPROP	Define common deck and propagation parameter.
*C	*COMPILE deckl,deck2, . . . ,deckn	Write specified decks to compile file, source file, and new program library.
	*COMPILE deckl,deck2	Write inclusive range of decks to compile file, source file, and new program library.
*CY	*COPY deck,c	Copy and insert specified card from named deck.
	*COPY deck,cl,c2	Copy and insert specified range of cards from named deck.
	*COPY deck,cl,c2,lfn	Copy specified range of cards from named deck to specified file.
*CW	*CWEOR level	Conditionally write end-of-record or end-of-file.
*DK	*DECK deck	Define deck to be included in program library.
*DC	*DECLARE deck	Following corrections restricted to named deck.
*DF	*DEFINE namel,name2, . . . ,namen	Defines names to be tested by IF directive while compile file is being written.
*D	*DELETE c	Deactivate specified card and optionally insert text in its place.
	*DELETE cl,c2	Deactivate inclusive range of cards and optionally insert text in their place.
none	*DO identl,ident2, . . . ,identn	Reactivate yanked cards in specified correction sets until a DONT is encountered.
*DT	*DONT identl,ident2, . . . ,identn	Terminate the DO for specified correction sets.
none	*END	Provides compatibility with the ENDITSYM program.
*EI	*ENDIF	Indicates end of conditional text.
*ET	*ENDTEXT	End delimiter for sequence of cards identifying test.
*ID	*IDENT idname,B=num,K=ident,U=ident	Define correction set, bias for seqnum, and whether specified correction sets must be known or unknown to process this set.
none	*IF type,name,num	Write specified number of following cards to the compile file if name of type DECK, IDENT, or DEF is known.
	*IF -type,name,num	Write specified number of following cards to the compile file if name is unknown.
*I	*INSERT c	Write subsequent text cards after card identified.

TABLE 3-1. SUMMARY OF UPDATE DIRECTIVES (Contd)

Directive Keyword Abbreviation	Directive Format	Use
*LT	*LIMIT n	Limit listable output to n lines.
*L	*LIST	Resume listing cards encountered in input stream.
*M	*MOVE deckl,deck2	Place deckl after deck2.
*NA	*NOABBREV	Do not check for abbreviated directives.
*NL	*NOLIST	Disable list option 4.
*PM	*PULLMOD identl,ident2, . . . ,identn	Recreate specified correction sets and write them to file specified by the G option.
*PD	*PURDECK deckl,deck2, . . . ,deckn	Permanently remove specified decks from program library.
	*PURDECK deckl.deck2	Permanently remove inclusive range of decks.
*P	*PURGE identl,ident2, . . . ,identn	Permanetly remove specified correction sets from program library.
	*PURGE identl.ident2	Permanetly remove inclusive range of correction sets.
	PURGE ident,	Permanently remove specified correction set and all sets introduced after it.
*RD	*READ lfn	Read directives and text from specified file.
*R	*RESTORE c	Reactivate specified card and optionally insert text after it.
	*RESTORE cl,c2	Reactivate inclusive range of cards and optionally insert text after them.
*RW	*REWIND lfn	Reposition named file to beginning-of-information.
*SP	*SELPURGE deckl.identl,deck2.ident2, . . . ,deckn.identn	Permanently remove all cards in specified deck that belong to specified correction set.
*SY	*SELYANK deckl.identl,deck2.ident2, . . . ,deckn-identn	Deactivate all cards in specified deck that belong to specified correction set.
*S	*SEQUENCE deckl,deck2, . . . ,deckn	Resequence all active cards and purge all inactive cards in specified decks.
	*SEQUENCE deckl.deck2	Resequence all active cards and purge all inactive cards in inclusive range of decks.
*SK	*SKIP lfn,n	Reposition named file forward specified number of logical records.
*T	*TEXT	Beginning delimiter for sequence of cards identifying text.
*W	*WEOR level	Write end-of-record or end-of-file according to specified level.
*Y	*YANK identl,ident2, . . . ,identn	Temporarily removes specified correction sets from program library.
	*YANK identl,ident2	Temporarily removes inclusive range of correction sets.
*YD	*YANKDECK deckl,deck2, . . . ,deckn	Temporarily deactivates decks specifeid.
none	*/ comment	Copy text to listable output file.

In the short form, idname is assumed to be the last explicitly named identifier given on a BEFORE, INSERT, DELETE, RESTORE, or COPY directive, whether or not it is a deck name. The dname is assumed to be the last explicitly named identifier given on a BEFORE, INSERT, DELETE, RESTORE, or COPY directive that is known to be a deck name. Both of these default identifiers are originally set to YANK$$$, therefore, the first directive using a card identifier must use the full form to reset the default.

All deck names are also identifiers (but all identifiers are not deck names). Thus, if EXAMPLE is the deck name last used, and there is no subsequent explicit reference to a correction set identifier, then both .281 and 281 expand to EXAMPLE.281 as the card identifier. If there is an explicit reference to a correction set identifier ABC after the explicit reference to the deck name, then 281 would expand to the card identifier ABC.281 while .281 would expand to EXAMPLE.281.

Figure 3-4 shows the differences in identifier expansion depending on the order of the directives. A is a deck name and B is a correction set identifier on an old program library.

```
*ID C
*INSERT A.2
     data card
*INSERT B.1
     data card
*D    2,  3            expands to *DELETE B.2, B.3
*D    4,  .5           expands to *DELETE B.4, A.5
*D   .7,  5            expands to *DELETE A.7, B.5
*D   .9,  .10          expands to *DELETE A.9, A.10

whereas:

*ID D
*INSERT B.1
     data card
*INSERT A.2
     data card
*D    2,  3            expands to *DELETE A.2, A.3
*D    4,  .5           expands to *DELETE A.7, A.5
*D   .7,  5            expands to *DELETE A.7, A.5
*D   .9,  .10          expands to *DELETE A.9, A.10
```

Figure 3-4. Examples of Card Identifier Expansion

DECK IDENTIFYING DIRECTIVES

Each deck to be placed on a program library must be introduced into the system by a DECK or COMDECK directive during a creation or correction run. When Update encounters one of these directives in the input stream prior to any correction directive, the run is considered to be a creation run. When Update encounters one of these directives while inserting new text cards, it terminates the insert and adds the decks to the program library following the card specified.

When a deck is added through the use of a DECK or COMDECK directive during a creation run or an ADDFILE directive during a correction run, termination of that deck occurs when Update encounters another DECK or COMDECK directive, or the end of a system-logical record. Cards within that deck are identified by the name of the deck or common deck to which they belong and are numerically sequenced beginning with 1 for the DECK or COMDECK directive. When a deck is inserted as text in a correction run (that is, through the use of an INSERT, DELETE, BEFORE, or RESTORE directive), it is terminated by any condition which normally terminates insertion. The contents of the deck, including the DECK or COMDECK card, are identified by the correction set name and are numerically sequenced as if they were normal insertion text.

Frequently, a DECK or COMDECK directive precedes each program or subprogram in a given program library. More than one subprogram, however, can be included in a deck, as is indicated in figure 3-5. Normally, two programs are grouped together if modification of one program requires reassembly of both programs.

```
*DECK       FIRST
            IDENT     FIRST
            . . . . .
            END
            IDENT     SECOND
            . . . . .
            END

*COMDECK    FDATA
            BLOCK DATA
            COMMON/J3/A(10)
            DATA A/3*0., 7*1.0/
            END
```

Figure 3-5. Example of Deck Structure

Because DECK and COMDECK directives can be deactivated by DELETE, YANK, or SELYANK, card images belonging to one deck at the beginning of an Update run can belong to a different deck at the end of the run. When a DECK or COMDECK directive is deactivated, all card images in the deactivated deck become members of the preceding deck on the program library; they retain their original card identifiers.

DECK DIRECTIVE

The DECK directive establishes a deck in the program library. It is one of the two directives that establishes the existence of a creation run. The directive can also be used in any correction run to add a deck to the location indicated by a preceding INSERT, BEFORE, DELETE, or RESTORE directive. Each deck must have a unique name within the program library. The DECK directive itself is part of the program library and has a sequence number of one within the name established by the directive. DECK directive format is shown in figure 3-6.

```
*DECK deck

deck      Name of deck.  Must be 1 through 9 characters.
          A through Z, 0 through 9, or + - / * ( ) $ =.
          Must not duplicate the name of any other deck
          in program library.
```

Figure 3-6. DECK Directive Format

COMDECK DIRECTIVE

The COMDECK directive establishes a common deck that can be called from other decks as they are being written to the compile file. It is one of the two directives that establishes the existence of a creation run. The directive can be used in any correction run to add a common deck to the location specified by a preceding INSERT, BEFORE, or

RESTORE directive. Each common deck must have a unique name. The COMDECK directive itself is part of the program library and has a sequence number of one within the name established by the directive. COMDECK directive format is shown in figure 3-7.

*COMDECK deck,NOPROP

deck — Name of deck. Must be 1 through 9 characters A through Z, 0 through 9, or + - / * ( ) $ =. Must notduplicate name of an existing deck.

NOPROP — Indicates that decks calling this common deck are not to be considered as modified when the common deck itself is modified; that is, the effects of common deck changes are not to be propagated during normal Update mode. Optional.

Figure 3-7. COMDECK Directive Format

The NOPROP parameter of the COMDECK directive determines whether a deck calling a corrected common deck is to be considered as having been corrected. If NOPROP is specified, only the common deck is considered to be corrected. On the other hand, if NOPROP is not specified, the common deck and the calling decks are considered to be corrected.

A common deck should be placed before any of the decks that call it. If the common deck is placed after a deck that calls it, Update may not be able to find it. In addition, decks calling a corrected common deck are not written to the compile file if the calling deck precedes the common deck and the mode is normal selective.

CORRECTION DIRECTIVES

Correction directives control updating of the old program library. New text is assigned a unique card identifier based on the correction set identifier. The corrected program library is written on the new program library; the old program library is not actually changed. Correction directives are illegal on a creation run.

ADDFILE DIRECTIVE

The ADDFILE directive causes Update to add a file of decks to the new program library. ADDFILE differs from the READ directive in that the contents of the specified file are limited to those allowed on a creation run. Unless the specified file is the primary input file, the READ directive cannot appear in the file. The first card image of the specified file must be a DECK or COMDECK directive. If the input file is specified, the READ directive can be the first image; a DECK or COMDECK directive must then be the first card image on the file specified by the READ directive. An ADDFILE directive cannot appear among directives read from the file specified by a READ directive. ADDFILE directive format is shown in figure 3-8. If only one parameter is specified, it is assumed to be lfn.

When the specified file is not the primary input file, Update adds directives and text until the end of one system-logical record is encountered. Update then returns to the file specified by the l parameter of the UPDATE control statement and continues processing the primary input stream. When the file specified on the ADDFILE directive is the primary input file, however, Update adds card images until a noncreation directive or the end of the system-logical record is encountered.

*ADDFILE lfn,name

lfn — Name of file from which decks are to be added. If lfn is omitted, the default is the file specified by the I parameter of the Update control statement; the separators are still required.

name — Name of deck or identifier of card after which decks are to be placed on the program library. If omitted, the addition is made after the last deck on the program library.

If the name parameter is *, it refers to the ident that is known to be a deck name most recently mentioned on a BEFORE, COPY, DELETE, INSERT, or RESTORE directive. If no such directive precedes the ADDFILE, YANK$$$ is used.

Figure 3-8. ADDFILE Directive Format

Update does not reposition the file specified on the ADDFILE directive. Any repositioning must be requested by the SKIP or REWIND directive.

BEFORE DIRECTIVE

The BEFORE directive inserts text card images and compile file directives in the program library before the specified card images. The card images to be inserted are placed immediately after the directive. Card images cannot be inserted into the YANK$$$ deck. The inserted card images receive card identifiers established by the correction set name of the preceding IDENT directive. BEFORE directive format is shown in figure 3-9.

*BEFORE c

c — Card identifier of card before which the insertion is to be made.

Figure 3-9. BEFORE Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURG, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they be text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

CHANGE DIRECTIVE

The CHANGE directive renames correction set identifiers. It cannot be used to change deck names. As a secondary effect, changing the name of the correction set invalidates any YANK or SELYANK directives that refer to the set by its previous name. Since a CHANGE directive goes into effect immediately, any subsequent references to the correction set must use the new name. The CHANGE directive need not be part of a correction set. CHANGE directive format is shown in figure 3-10.

```
*CHANGE oldid,newid, . . . ,oldid,newid

oldid    Name of correction set to be changed.

newid    New correction set name. Must be 1 through 9
         characters A through Z, 0 through 9, or + - / *
         ( ) $ =. Must not duplicate the name of any
         other correction set in the program library.
```

Figure 3-10. CHANGE Directive Format

COPY DIRECTIVE

The COPY directive copies active card images from a deck on the old program library and inserts the images into another deck as if they were text in an input stream, or the COPY directive copies active card images to a specified file. Since Update copies the card images into a deck before applying corrections to them, card images can be copied and original images can be modified in the same run. An attempt to copy card images introduced during the same Update run produces an informative message. COPY directive format for copying card images to a deck on the program library is shown in figure 3-11. COPY directive format for copying card images to a file is shown in figure 3-12.

```
A.  Copy specified card.

    *COPY deck,c

    deck    Name of deck on old program library that
            contains the card to be copied.

    c       Card identifier of card to be copied.

B.  Copy range of cards.

    *COPY deck,c1,c2

    deck    Name of deck on old program library that
            contains cards to be copied.

    c1,c2   Card identifiers of first and last cards in
            sequence of cards to be copied.
```

Figure 3-11. COPY Directive Format – Copy to Deck

An INSERT, DELETE, BEFORE, or RESTORE directive must be in effect to use COPY to copy card images to a deck. In figure 3-13A, the use of the COPY directive is valid because a preceding INSERT directive has initiated insertion. Card images BDECK.4 through BDECK.8 are copied and inserted after the text cards. The copied card images are sequenced as part of correction set X. The input stream in figure 3-13B is not valid because insertion is not in effect to indicate where to write the card image copies.

Placement in the input stream of a COPY directive that copies card images to a file is not restricted; COPY can appear anywhere in the primary input stream. Copying card images to a file is illegal, however, when a secondary input stream is being read as a result of a READ directive.

```
*COPY deck,c1,c2,lfn

deck    Name of deck on old program library that con-
        tains cards to be copied.

c1,c2   Card identifiers of first and last cards in
        sequence of cards to be copied.

lfn     Name of file onto which cards are to be copied.
        The user is responsible for the disposition of
        this file since it is not positioned either before
        or after the copy. The file is written as a
        coded file that contains 80-column card images
        with one system-logical record for each COPY
        directive; sequencing information is not included.
```

Figure 3-12. COPY Directive Format – Copy to File

```
A.  Valid use of COPY.

    *IDENT X
    *INSERT BLAP.11
    (text cards)
    *COPY BDECK,BDECK.4,BDECK.8

B.  Invalid use of COPY.

    *IDENT X
    *COPY BDECK,BDECK.4,BDECK.8
```

Figure 3-13. Example of Use of COPY

DELETE DIRECTIVE

The DELETE directive deactivates a card image or a group of card images and optionally inserts text and directives after the deleted card images. The card images to be inserted are placed immediately after the directive. The inserted card images receive card identifiers established by the correction set name of the preceding IDENT directive. DELETE directive format depends on whether card images to be deactivated are specified by card identifier or by a range of cards, as shown in figure 3-14.

```
A.  Delete specified card

    *DELETE c

    c       Card identifier for single card to be deleted.

B.  Delete range of cards

    *DELETE c1,c2

    c1,c2   Card identifiers of first and last cards,
            in sequence of cards to be deleted. Card
            c1 must appear before c2 in the existing
            library. The range can include cards already
            in a deactivated state.
```

Figure 3-14. DELETE Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they be text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

IDENT DIRECTIVE

The IDENT directive establishes the name for the set of corrections being made. Cards added in this correction set are sequenced within the name specified. All correction set names must be unique. If a new program library is not being generated, a correction set need not begin with an IDENT directive. In this case, Update uses the default name of .NO.ID. for new text cards. The established correction set identifier remains in effect until Update encounters another IDENT directive or a PURGE, SELPURGE, PURDECK, ADDFILE, or SEQUENCE directive. IDENT directive format is shown in figure 3-15.

```
*IDENT idname,B=num,K=ident,U=ident

idname    Name to be assigned to this correction set.
          Must be 1 through 9 characters A through
          Z, 0 through 9, or + - / * ( ) $ =. Must
          not duplicate the name of another correction
          set or deck. This directive causes a new entry
          in the directory.

B=num     Bias to be added to sequence numbers within
          deck. Optional.

K=ident   Indicator that specified correction set name
          must exist in the directory of the library
          before corrections can be made. Optional.

U=ident   Indicator that specified correction set name
          must not exist in the directory of the library.
          Optional.
```

Figure 3-15. IDENT Directive Format

Omitting idname causes a format error. If idname duplicates a name previously used, Update issues an error message. Both errors are nonfatal as long as no new program library is created in the same run.

The B, K, and U parameters on the IDENT directive can appear in any order. If more than one B parameter is specified, Update uses the last one encountered. More than one K or U parameter can be specified; in this instance, all correction set names must be known or unknown as specified before the correction set is processed. (An identifier is known whether it is active or inactive; an identifier that has been yanked is still known. To become unknown, an identifier must be purged.) If the criteria of these parameters is not met, Update skips the correction set and resumes processing with the next IDENT, PURGE, SELPURGE, PURDECK, or ADDFILE directive.

In the following example, the bias of 100 is added to all ZAP correction set card sequence numbers:

```
    *IDENT ZAP,B=100,K=ACE,U=NON,U=ARF
```

The first card image in correction set ZAP has a sequence number of 101, not 1. Update skips the correction set if ACE is unknown or either NON or ARF is known.

INSERT DIRECTIVE

The INSERT directive inserts text card images and compile file directives in the program library after the specified card image. The card images to be inserted are placed immediately after the directive. Card images cannot be inserted into the YANK$$$ deck. The inserted card images receive card identifiers established by the correction set name of the preceding IDENT directive. INSERT directive format is shown in figure 3-16.

```
*INSERT c

c     Card identifier of card after which insertion is to
      be made.
```

Figure 3-16. INSERT Directive Format

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURGE, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they be text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the text.

MOVE DIRECTIVE

The MOVE directive enables the user to reorder decks while producing a new program library. The deck to be repositioned is moved from its position on the old program library and placed after the specified deck on the new program library. The YANK$$$ deck cannot be moved. A MOVE referencing a deck introduced in the same Update run produces an informative message. This directive does not terminate insertion and need not be part of a correction set. MOVE directive format is shown in figure 3-17.

```
*MOVE deck1,deck2

deck1    Deck name on old program library to be moved.

deck2    Deck name after which deck1 is to be placed
         on new program library.
```

Figure 3-17. MOVE Directive Format

PURDECK DIRECTIVE

The PURDECK directive permanently removes a deck or group of decks from the program library. The YANK$$$ deck cannot be purged. Every card image in the deck is purged, regardless of what correction set it might belong to. Purging, unlike yanking, cannot be rescinded. A PURDECK directive can appear anywhere in the input stream, its appearance terminates the current correction set. PURDECK directive format depends on whether decks to be purged are specified individually by deck name or by a range of deck names, as shown in figure 3-18.

The name of a purged deck is removed from the deck list; it can be reused as a deck name. An entry for the purged deck remains in the directory, however, until removed through the use of the E parameter on the UPDATE control statement. The deck name can also be removed from the directory by resequencing the library, that is, by creating a source file in one Update run and then using the source file as input on a subsequent creation run. Until a deck name is removed from the directory, it cannot be used as a correction set identifier.

A. Purge decks listed

*PURDECK deck1,deck2, . . . ,deckn

deck — Name of deck to be purged. Names can appear in any order.

B. Purge range of decks

*PURDECK deck1.deck2

deck1.deck2 — Names of first and last decks, inclusive, to be purged. Names must appear in the relative order in which decks exist in the deck list.

Figure 3-18. PURDECK Directive Format

PURGE DIRECTIVE

The PURGE directive permanently removes a correction set or group of correction sets from the program library. Every card in the correction set is purged, regardless of its status as active or inactive. Purging, unlike yanking, cannot be rescinded. A new program library written during the same run will treat the purged correction set as if it had never existed. A PURGE directive can appear anywhere in the input stream; it terminates the current correction set. PURGE directive format, as shown in figure 3-19, depends on whether correction sets to be purged are specified individually by correction set name, by a range of correction set names, or by relative time of introduction into the program library.

If Update cannot locate a specified correction set, it issues an error message. Purged identifiers can be reused on subsequent correction sets provided they do not appear in the YANK$$$ deck as a YANK directive parameter.

RESTORE DIRECTIVE

The RESTORE directive reactivates a card image or a group of card images previously deactivated through a DELETE directive. Any text card images and compile file directives immediately following the RESTORE directive are inserted after the last card image identified on the directive. Any inserted card images receive card identifiers established by the correction set name of the preceding IDENT directive. RESTORE directive format depends on whether card images to be reactivated are specified by card identifier or by a range of cards, as shown in figure 3-20.

Unless a TEXT directive has been encountered, Update terminates an insertion when it encounters the next insertion directive or a PURGE, PURDECK, IDENT, SELPURG, ADDFILE, or SEQUENCE directive. On the other hand, compile file directives are inserted as if they be text after Update checks for correct syntax. Update interprets all other directives without terminating insertion; however, the directives are not inserted into the deck.

A. Purge listed correction sets

*PURGE ident1,ident2, . . . ,identn

ident — Identifier of a correction set to be purged. Identifiers can appear in any order.

B. Purge range of correction sets

*PURGE ident1.ident2

ident1.ident2 — Identifiers of first and last correction sets, inclusive, to be purged. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order they exist in the directory.

C. Purge later correction sets

PURGE ident,

ident — Identifier of correction set to be purged along with all correction sets introduced after the specified correction set.

* — Indicator that the program library is to return to an earlier level. Intervening purge directives and SEQUENCE prevent complete return.

Figure 3-19. PURGE Directive Format

A. Restore specified card.

*RESTORE c

c — Card identifier of card to be restored.

B. Restore range of cards.

*RESTORE c1,c2

c1,c2 — Card identifiers of first and last cards, inclusive, in sequence of cards to be restored. Card c1 must appear before c2 in the existing library. Any cards in the sequence that are already active are not affected.

Figure 3-20. RESTORE Directive Format

SELPURGE DIRECTIVE

The SELPURGE directive permanently removes the effects of the specified correction set on the specified deck. Only the card images belonging to the specified correction set are purged from the specified deck. Card images belonging to the specified correction set that are in other decks are not purged. Card images in the YANK$$$ deck can be purged thorugh SELPURGE. A SELPURGE directive can appear anywhere in the input stream; it terminates the current correction set. SELPURGE directive format is shown in figure 3-21.

*SELPURGE deck1.ident1, . . . ,deckn.identn

deck Name of deck from which correction set is to be removed.

ident Name of correction set to which cards to be removed belong. It must be separated from the deck by a period.

Figure 3-21. SELPURGE Directive Format

SELYANK DIRECTIVE

The SELYANK directive termporarily removes the effects of the specified correction set on the specified deck. Only the card images belonging to the specified correction set are yanked from the specified deck. Card images belonging to the specified correction set that are in other decks are not yanked. Card images in the YANK$$$ deck can be yanked through SELYANK. A SELYANK directive must be part of a correction set; it is placed in the YANK$$$ deck. SELYANK directive format is shown in figure 3-22.

*SELYANK deck1.ident1, . . . ,deckn.identn

deck Name of deck from which correction set is to be removed.

ident Name of correction set to which cards to be removed belong. It must be separated from deck by a period.

Figure 3-22. SELYANK Directive Format

SEQUENCE DIRECTIVE

The SEQUENCE directive resequences active cards and purges inactive cards from the specified decks. Only those decks explicitly mentioned on the SEQUENCE directive are resequenced. Thus, if a correction set (for example, SET1) affects more than one deck on a program library (for example, DECK1 and DECK2), and only DECK1 has been subsequently resequenced through SEQUENCE, the SEQUENCE directive does not affect SET1 cards within DECK 2. The YANK$$$ deck cannot be resequenced. SEQUENCE directive format, as shown in figure 3-23, depends on whether decks to be resequenced are specified individually by name or are specified as a range of decks names.

Update normally allows deck and correction sets having the same name to coexist on the old program library. If a deck having the same name as a correction set is resequenced and cards for the correction set are in other decks, Update purges any modifications made by that correction set outside the resequenced deck to prevent duplicate identifiers.

SEQUENCE does not result in identifiers being deleted from the directory even if, as a result of resequencing, no references to an identifier are on the library. This situation arises when all the corrections of a correction set refer to a deck that is resequenced. Deletion of the identifier, in this case, requires an edit or PURGE in a subsequent Update run.

A deck cannot be renamed and resequenced in the same Update run. (To rename a deck, delete the first card of the deck and replace it with a new DECK directive containing the new name.)

A. Resequence listed decks.

*SEQUENCE deck1,deck2, . . . ,deckn

deck Name of deck to be resequenced.

B. Resequence range of decks.

*SEQUENCE deck1.deck2

deck1.deck2 Name of first and last decks, inclusive, to be resequenced. Deck1 must appear before deck2 in old program library.

Figure 3-23. SEQUENCE Directive Format

YANK DIRECTIVE

The YANK directive temporarily removes a correction set or group of correction sets from the program library. Card images activated by the correction set are deactivated; card images deactivated by the correction are reactivated. If a correction set has been yanked, it is ignored during compile file or source file generation. The effects of an yank can be selectively nullified through the introduction of DO and DONT directives in the decks. Update places the YANK directive in the YANK$$$ deck. YANK directive format, as shown in figure 3-24, depends on whether correction sets to be yanked are specified individually by correction set name or by a range of correction set names.

A. Yank listed correction sets

*YANK ident1,ident2, . . . ,identn

ident Identifier of a correction set to be yanked. Identifiers can appear in any order.

B. Yank range of correction sets

*YANK ident1.ident2

ident1.ident2 Identifiers of first and last correction sets, inclusive, to be yanked. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order they exist in the directory.

Figure 3-24. YANK Directive Format

The YANK directive differs from PURGE in several respects. YANK must be part of a correction set. YANK does not terminate the current correction set. And, the effects of a YANK directive can be rescinded.

YANKDECK DIRECTIVE

The YANKDECK directive temporarily removes all cards within the decks specified. All cards are deactivated, even if they belong to a correction set. YANKDECK differs from PURDECK in several respects: YANKDECK must be part of a correction set; it does not terminate the current correction set; and its effects can be rescinded. YANKDECK directive format is as shown in figure 3-25.

```
*YANKDECK deck1,deck2, . . .,deckn

deck   Name of deck to be yanked.  Names can appear
       in any order.
```

Figure 3-25. YANKDECK Directive Format

The deck YANK$$$ cannot be deactivated as a whole. Individual YANK directives within this deck can be yanked by a YANK directive, however.

COMPILE FILE DIRECTIVES

Compile file directives provide control over the compile file. These directives are interpreted when the program library decks are being corrected and written onto the compile file. Calls for common decks result in the common deck being written on the compile file. Other directives allow control of file format. None of the compile file directives are written on the compile file.

The user can prepare the original deck with embedded compile file directives (except for DO or DONT) or the user can insert compile file directives into program library decks as a part of a correction set. Compile file directives are not processed when they are encountered in the input stream (except for COMPILE); they are simply considered as text cards to be inserted and are sequenced accordingly after update checks for correct syntax. To be recognized while the compile file is being written, these directives must have the same master control character as defined when the library was created.

CALL DIRECTIVE

The CALL directive causes the active text of a common deck to be written onto the compile file. The directive itself is stored as part of a deck and can be referenced by its card identifier. CALL is effective only within a deck or common deck. Common decks can call other common decks, but a common deck must not either call itself or call a common deck that contains a call to the common deck. Neither the CALL directive nor the COMDECK directive which defined the deck is written to the compile file. COMDECK directive format is shown in figure 3-26.

```
*CALL deck

deck   Name of an existing common deck to be written
       to the compile file.
```

Figure 3-26. CALL Directive Format

Common decks can also be called from secondary old program libraries. If COMDECK names are duplicated on any secondary old program libraries, Update uses the first COMDECK encountered according to the order of the secondary old program libraries as specified by the P parameter of the UPDATE control statement.

COMPILE DIRECTIVE

The COMPILE directive indicates which decks are to be written to the compile file during normal or quick Update mode. The directive is ignored during a full Update.

Normal mode — Decks specified on COMPILE directives and corrected decks are written to the compile file.

Quick mode — Decks specified on COMPILE directives and any common decks they call are written to the compile file.

The directive also affects the contents of any new program library and source file as shown in table 2-2. COMPILE directive format, as shown in figure 3-27, depends on whether decks to be written are specified individually by name or are specified as a range of deck names.

```
A.  Compile listed decks

    *COMPILE deck1,deck2, . . . ,deckn

    deck          Name of deck to be written to the
                  compile file, new program library file,
                  and source file.

B.  Compile range of decks

    *COMPILE deck1.deck2

    deck1.deck2   Names of first and last decks in range,
                  inclusive, to be written to the compile
                  file.  The name of deck1 must appear
                  before the name of deck 2 in the old
                  program library deck list.
```

Figure 3-27. COMPILE Directive Format

Decks are written to the compile file in the order that the decks exist on the old program library, unless the K option is selected on the UPDATE control statement. If the K option has been specified, the decks are written in the order they appear on the COMPILE directive.

When a deck is being introduced in the same run that contains a COMPILE directive for the deck, the DECK directive must appear before the COMPILE directive. Otherwise, COMPILE directives can be anywhere in the input stream. They do not affect the current correction set name.

CWEOR DIRECTIVE

The CWEOR directive causes the termination of the current system-logical record on the compile file with the specified level only if information has been placed in the output buffer since the last system-logical record was written. CWEOR directive format is shown in figure 3-28.

```
*CWEOR level
```

level Level of system-logical record.

For SCOPE 2, the following:

RT=W	0 thru 14	end-of-section
RT=W	15	end-of-partition
RT=S	0 thru 15	end-of-record
RT=Z	0 thru 15	end-of-section
BT=C	0 thru 15	end-of-section

Figure 3-28. CWEOR Directive Format

DO DIRECTIVE

The DO directive causes Update to rescind a yank of specified correction sets while writing text to the compile file. If a card was deactivated as a result of a YANK or SELYANK, the card is reactivated. Likewise, if a card was activated by a YANK or SELYANK, Update deactivates it. A DO remains in effect until a DONT directive is encountered. The DO directive can be placed anywhere in the library. If Update encounters a DO for an unyanked correction set, an informative message is issued and the DO is ignored. DO directive format is as shown in figure 3-29.

```
*DO ident1,ident2, . . . ,identn
```

ident Name of correction set for which yanking is to be rescinded or initiated.

Figure 3-29. DO Directive Format

DONT DIRECTIVE

The DONT directive terminates a DO directive. It can also be used to initiate a yank of an unyanked correction set. When Update encounters a DONT for a correction set that has not been yanked, it yanks the set until it encounters a DO directive for the set. If the correction set has already been yanked, Update issues an informative message and ignores the DONT. The DONT directive can be placed anywhere in the program library. DONT directive format is as shown in figure 3-30.

```
*DONT ident1,ident2, . . .,identn
```

ident Name of correction set for which yanking is to be rescinded or initiated.

Figure 3-30. DONT Directive Format

ENDIF DIRECTIVE

The ENDIF directive indicates the end of conditional text. It is used with IF when the num parameter is omitted from the IF directive. ENDIF should not be used if num is specified on the IF directive. Since num takes precedence, the ENDIF directive is included in the count of active cards and is written on the compile file. ENDIF directive format is shown in figure 3-31.

```
*ENDIF
```

Figure 3-31. ENDIF Directive Format

IF DIRECTIVE

The IF directive conditionally writes text on the compile file. When Update encounters an IF directive, the text following the directive is written or skipped depending on the condition. IF directive format, as shown in figure 3-32, depends on whether the specified name is to be known or unknown for the text to be written on the compile file.

A. Name must be known.

```
*IF type,name,num
```

B. Name must be not known.

```
*IF -type,name,num
```

type Type of condition name.

DECK Name is deck name. To be known, it must be in the deck list on the primary old program library.

IDENT Name is correction set identifier. To be known, it must be in the directory on the primary old program library.

DEF Name is defined through DEFINE directive on the old program library.

When type is not preceded by a minus sign, the name must be known for text to be written. When type is preceded by a minus sign, the name must not be known for text to be written.

name Deck name, correction set identifier, or defined name, according to type.

num Number of active card images to be skipped if condition is not met. Optional.

Figure 3-32. IF Directive Format

If the num parameter is omitted and the condition is not met, Update searches for an ENDIF directive and resumes processing of the deck at that point. When the condition is met, no cards are skipped.

When an IF directive is encountered on a secondary old program library, Update will only search the directory, deck list, and YANK$$$ deck on the primary old program library in trying to satisfy the conditional. The deck lists, directories, and YANK$$$ decks of the secondary old program libraries are not searched.

When both an IF directive is encountered as a result of a CALL and a matching ENDIF directive is found as the result of a second CALL, the range of the IF, ENDIF pair is unpredictable.

WEOR DIRECTIVE

The WEOR directive causes the termination of the current system-logical record on the compile file with the specified level. WEOR directive format is shown in figure 3-33.

```
*WEOR level

level   Level of system-logical record.

        For SCOPE 2, the following:

        RT=W   0 thru 14    end-of-section
        RT=W   15           end-of-partition
        RT=S   0 thru 15    end-of-record
        RT=Z   0 thru 15    end-of-section
        BT=C   0 thru 17    end-of-section
```

Figure 3-33. WEOR Directive Format

FILE MANIPULATION DIRECTIVES

File manipulation directives control secondary input files during Update processing. These directives can only appear in the primary input stream. They are illegal on a secondary input file.

READ DIRECTIVE

The READ directive temporarily stops reading the primary input stream and begins reading an input stream from the specified file. READ differs from ADDFILE in that the content of the file specified by READ is not restricted except to prohibit the appearance of another READ directive or the ADDFILE, SKIP, and REWIND directives. Update reads from the specified file one system-logical record. Processing then continues with the main input stream. READ directive format is shown in figure 3-34.

```
*READ lfn

lfn     Name of alternate file containing input stream.
```

Figure 3-34. READ Directive Format

The specified file cannot be one of the reserved files specified by a parameter on the UPDATE control statement. It can only be a secondary input file.

REWIND DIRECTIVE

The REWIND directive repositions the specified file to beginning-of-information. The file to be rewound cannot be one of the reserved files. It can only be a secondary input file. REWIND directive format is shown in figure 3-35.

```
*REWIND lfn

lfn     Name of file to be rewound.
```

Figure 3-35. REWIND Directive Format

SKIP DIRECTIVE

The SKIP directive repositions the named file forward one or more system-logical records. A system-logical record of level 17_8 or end-of-information terminates skipping. SKIP directive format is shown in figure 3-36.

```
*SKIP lfn,n

lfn     Name of file to be positioned.

n       Number of logical records to be skipped in the
        forward direction. If n is omitted, Update skips
        one record.
```

Figure 3-36. SKIP Directive Format

INPUT STREAM CONTROL DIRECTIVES

The input stream control directives allow the user to specify whether or not Update is to recognize abbreviated directives, delimit text, or control which input stream cards are to be listed on the output file.

ABBREV DIRECTIVE

The ABBREV directive causes checking for abbreviated directives to be resumed. It is used in connection with the NOABBREV directive. ABBREV directive format is shown in figure 3-37.

```
*ABBREV
```

Figure 3-37. ABBREV Directive Format

ENDTEXT DIRECTIVE

The ENDTEXT directive ends the condition established by a prior text directive. If ENDTEXT is encountered before TEXT, Update ingores it. ENDTEXT directive format is shown in figure 3-38. Any information in columns 10 through 80 is taken as a comment.

```
*ENDTEXT
```

Figure 3-38. ENDTEXT Directive Format

LIST DIRECTIVE

The LIST directive causes listing of cards in the input stream to be resumed. It is used in connection with NOLIST. LIST directive format is as shown in figure 3-39.

```
*LIST
```

Figure 3-39. LIST Directive Format

NOABBREV DIRECTIVE

The NOABBREV directive causes Update to stop checking for the abbreviated forms of the directives. Update expands the name when it reads an abbreviated form so that it is a full name. Because checking for the abbreviated forms and expanding them is a time-consuming feature, the user has the option of not using abbreviations and of turning off the check through the NOABBREV feature. In this mode, an abbreviated directive is not recognized but is taken as text. NOABBREV directive format is shown in figure 3-40.

```
*NOABBREV
```

Figure 3-40. NOABBREV Directive Format

NOLIST DIRECTIVE

The NOLIST directive disables list option 4. Update stops listing cards in the input stream when it encounters a NOLIST and resumes listing cards when it encounters a LIST. NOLIST directive format is shown in figure 3-41.

```
*NOLIST
```

Figure 3-41. NOLIST Directive Format

LIST and NOLIST can occur anywhere in the input stream. They do not terminate insertion or a correction set. The LIST/NOLIST directives are ignored if list option 0 is selected.

TEXT DIRECTIVE

The TEXT directive, used in connection with ENDTEXT, causes all following card images to be treated as text, whether or not they begin with the master control character and would otherwise be considered as directives. When Update encounters a TEXT directive, that card image and all following it up to and including the ENDTEXT directive are considered as text and are written on the program library. A TEXT directive in the input stream must be either in a deck or in text being inserted. The TEXT and ENDTEXT directives are maintained on the program library as text card images; however, they are not written on the compile file. TEXT format is shown in figure 3-42. Any information in columns 10 through 80 is taken as a comment.

```
*TEXT
```

Figure 3-42. TEXT Directive Format

SPECIAL DIRECTIVES

The special directives provide extended features. With the exception of DEFINE and PULLMOD, they can appear any place in the input stream for creation or correction runs.

DELCARE DIRECTIVE

The DECLARE directive protects decks other than the declared deck from being inadvertently altered. Subsequent corrections are restricted to the named deck until Update encounters a DECLARE directive with no deck name or another DECLARE directive with a different deck name. This directive can only be used when the DECLKEY installation option has been assembled. DECLARE directive format is shown in figure 3-43.

```
*DECLARE deck

deck   Name of deck to which following corrections are
       restricted.
```

Figure 3-43. DECLARE Directive Format

When the DECLARE directive is encountered, the following restrictions go into effect:

PURGE and YANK directives are illegal.

INSERT, DELETE, RESTORE, and BEFORE directives can apply only to cards in the declared deck. If they do not, the operation is not performed and Update issues an informative message.

Inserting or reactivating a DECK or COMDECK directive is illegal.

New decks inserted via the ADDFILE directive need not be named in a DECLARE directive.

DEFINE DIRECTIVE

The DEFINE directive establishes a condition to be tested by the IF directive. The names on a DEFINE directive are unrelated to correction set identifiers or deck names. Update places DEFINE directives in the YANK$$$ deck. A DEFINE directive can be placed anywhere in a correction set. DEFINE directive format is shown in figure 3-44.

```
*DEFINE name1,name1, . . . ,namen

name   Name for subsequent testing by IF directive.
```

Figure 4-44. DEFINE Directive Format

END DIRECTIVE

The END directive provides compatibility with the EDITSYM program. Update ignores an END directive if it encounters one in a deck. Update does not copy the END directive onto the program library. END directive format is shown in figure 3-45.

```
*END
```

Figure 4-45. END Directive Format

LIMIT DIRECTIVE

The LIMIT directive changes the maximum size for the listable output file from the default value of 6000 lines to the specified number of lines. It should be one of the first cards encountered in the input stream. The LIMIT directive will not appear in the new program library. LIMIT directive format is shown in figure 3-46.

```
        *LIMIT n

        n       New line limit for listable output.
```

Figure 4-46. LIMIT Directive Format

When the specified limit is reached, options 3 (card image, deck name, and modification key) and 4 (input stream) are turned off. Errors and directives are still listed, however, if options 1 and 2 were selected. Options 5 through 9 are not affected. Refer to L parameter in section 4.

PULLMOD DIRECTIVE

The PULLMOD directive causes the program library to be searched for all card images belonging to each specified correction set and reconstructs a set of directives and text. The reconstructed correction set produces the same results as the original set. The search of the library is performed at the end of the Update run. Therefore, any modifications made by the current run are reflected in the PULLMOD results. Each reconstructed correction set is written to the file specified by the G parameter on the UPDATE control statement. All of the sets are contained within one system-logical record on the file. PULLMOD directive format is shown in figure 3-47. The PULLMOD directive can be used only when the PMODKEY installation option has been assembled for Update.

The user is responsible for determining whether or not the reconstructed correction sets accurately reflect the original corrections. PULLMOD is unable to determine if card images have been purged subsequent to the addition of the correction sets requested.

```
        *PULLMOD ident1,ident2, . . . ,identn

        ident   Name of correction set to be recreated.
```

Figure 4-47. PULLMOD Directive Format

A pullmod file has the same format as an input file. This feature permits a user to take an earlier version of the library and apply selected correction sets.

/ COMMENT DIRECTIVE

The / directive introduces a comment into the listable output file. Update ignores this card except to copy it to the output file. A comment can appear at any place in the input stream. The slash can be redefined as another character through the / parameter of the UPDATE control statement. The / comment directive format is shown in figure 3-48. The slash must appear in column 2. Column 3 must be a comma or blank.

```
     */ comment
```

Figure 3-48. Comment Directive Format

UPDATE CONTROL STATEMENT 4

The Update utility is called by the UPDATE control statement. Parameters specify options and files for the run. The format of the call is shown in figure 4-1. The word UPDATE must begin in column one. See the operating system reference manual for additional control statement syntax requirements.

```
UPDATE(p-list)
```

p-list Parameters specifying options. Parameters in the list are separated by commas. A left parenthesis or a comma must separate the list from the word UPDATE. A right parenthesis or a period terminates the statement.

Figure 4-1. UPDATE Control Statement Format

PARAMETERS

All Update parameters are optional and can appear in any order. Parameters are summarized in table 4-1. They are described in detail below.

A SEQUENTIAL-TO-RANDOM COPY

This parameter copies a sequential old program library to a random new program library. No other Update operations are performed; any I parameter is ignored. The only other control statement parameters that can be used with the A parameter are those specifying files, L=0, R, *, and /. An error results if the old program library is not sequential or the new program library is not random. For SCOPE 2, the new program library cannot be blocked.

omitted	No copy made.
A	The sequential old program library is copied to a random new program library.

B RANDOM-TO-SEQUENTIAL COPY

This parameter copies a random old program library to a sequential new program library. No other Update operations are performed; any I parameter is ignored. The only other control statement parameters that can be used with the B parameter are those specifying files, L=0, R, *, and /. An error results if the old program library is not in random format.

omitted	No copy made.
B	The random old program library is copied to a sequential new program library.

C COMPILE FILE NAME

This parameter specifies the name of the compile file. The content of the compile file is determined by the Update mode as shown in table 2-2.

omitted or C	Decks are written to the file named COMPILE.
C=lfn	Decks are written to file named lfn.
C=PUNCH	Decks are written to file named PUNCH. The D and 8 parameters are implied.
C=0	Compile file suppressed.

The C parameter is ignored if K is also specified

D DATA WIDTH ON COMPILE FILE

This parameter specifies how many columns are to be used for data on the COMPILE file. Data width does not include sequencing information.

omitted	72 columns of data
D	80 columns of data

E EDIT OLD PROGRAM LIBRARY

This parameter specifies that the old program library is to be edited. During editing, the directory and deck list are rearranged to reflect the actual order of decks on the program library; all previously purged identifiers are removed. Identifiers that exist simply as entries in the directory and have no cards associated with them are purged. Any cards other than YANK, SELYANK, YANKDECK, or DEFINE that exist in the YANK$$$ deck are also purged.

Two edit runs are required to edit the library completely. The first edit run removes purged identifiers and flags unused identifiers as purged. The second edit run deletes the unused identifiers from the directory.

omitted	No editing is done.
E	The program library is edited.

The E parameter can only be used when the EDITKEY installation option has been assembled for Update.

F FULL UPDATE MODE

This parameter specifies full Update mode.

omitted	Normal selective Update mode, as long as Q is not specified.
F	Full Update mode.

G PULLMOD FILE NAME

This parameter specifies the name of the pullmod file.

omitted	Output from PULLMOD directives is appended to the source file.
G=lfn	Output from PULLMOD directives is written on file named lfn. The listable output file (O parameter) cannot be specified.

The G parameter can only be used when the PMODKEY installation option has been assembled for Update.

TABLE 4-1. SUMMARY OF UPDATE CONTROL STATEMENT PARAMETERS

Parameter	Function
A	Copy sequential old program library to new random program library.
B	Copy random old program library to new sequential old program library.
C	Specify name of compile file.
D	Define compile file card image width excluding Update sequence information.
E	Remove from directory previously purged identifiers and purge identifiers that exist Simply as directory entries.
F	Select full update mode.
G	Specify name of Pullmod file.
H	Specify character set.
I	Specify name of file with input stream.
K	Write decks on compile file in order specified on COMPILE directives.
L	Select listable output file contents.
M	Merge specified program library with old program library.
N	Specify name of new program library file.
O	Specify name of listable output file; content is determined by L parameter.
P	Specify names of old program library and secondary old program libraries.
Q	Select quick update mode.
R	Rewind specified files.
S	Specify name of source file; content includes common decks and is determined by mode.
T	Same as S, but omit common decks.
U	Do not terminate execution if fatal error occurs.
W	Specify sequential new program library file.
X	Specify compressed format for compile file.
8	Define compile file card image width including Update sequence information.
*	Redefine master control character for directives.
/	Redefine control character for comments.

H CHARACTER SET CHANGE

This parameter allows the user to override the character set type specification in the old program library.

omitted or H — Update treats the old program library character set as the character set indicated in the old program library.

H=3 — Update treats the old program library as a 63-character set program library regardless of the character set specified in the old program library.

H=4 — Update treats the old program library as a 64-character set program library regardless of the character set specified in the old program library.

I INPUT STREAM FILE NAME

This parameter specifies the name of the primary input file.

omitted or I — Directives and text are on the file named INPUT.

I=lfn — Directives and text are on file named lfn.

K COMPILE FILE SEQUENCE

This parameter specifies that decks are to be written to the Compile file in the order in which the deck names are encountered on COMPILE directives. If a deck name is mentioned more than once, its last specification determines the deck's place within the compile file.

This parameter takes precedence over the C parameter.

omitted — Location determined by C parameter.

K — Decks to be written on file named COMPILE in COMPILE directive sequence.

K=lfn — Compile output decks to be written on file named lfn in COMPILE directive sequence.

L LISTABLE OUTPUT OPTIONS

This parameter specifies the content of the output file.

omitted — For a creation run, selects options A, 1, and 2.

For a correction run, selects options A, 1, 2, 3, and 4.

For a copy run, selects options A and 1.

L=c. . .c — Each character in string c. . .c selects one of the following options. The character 0 overrides any other options specified and suppresses the entire listing.

A List known deck names and correction set identifiers, COMDECK directives that were processed, known definitions (DEFINE directive), and decks written to the compile file.

F All options except 0.

0 All listing is suppressed.

1 List cards in error and the associated error messages. The flag *ERROR* appears to the left and right of an erroneous card image.

2 List all active Update directives encountered either on the input file or on the old program library. Those directives encountered in input are flagged with five asterisks to the left unless the directive is abbreviated or the card identifier is in short form. In this case, the directive is flagged with five slashes. If the directive has been encountered on the old program library, the name of the deck to which this card belongs is printed in palce of the five asterisks or slashes.

3 Comment on each card that changed status during current run. Comments include the deck name, card image, card identifier, and an indicator of action taken for that card.

I Card added.

A Inactive card reactivated.

D Active card deactivated.

P Card purged. If the card was active, ACTIVE also appears.

SEQ Card resequenced.

4 List text cards encountered in the input stream. Cards read as a result of a READ directive are identified to the right with the file name. Cards inserted as a result of an ADDFILE directive are listed only when option 4 is explicitly selected. Cards inserted as a result of a COPY directive are identified to the right by the word copy.

Option 4 may be turned on by a LIST directive and off by a NOLIST directive.

5 List all active compile file directives.

6 List number of active and inactive cards by deck name and correction set identifier.

7 List all active cards; identify to the right with an A.

8 List all inactive cards; identify to the right with an I.

9 List correction history of all cards selected by list options 5, 7, and 8.

List options 5 through 9 are provided for auditing an old program library. These options are available only when the AUDITKEY installation option is assembled. Output is written to a temporary file and appended to the listable output file at the end of the Update run. When the F parameter is selected, options 5 through 9 apply to all decks on the old program library. If F is not selected, options 5 through 9 apply to decks listed on COMPILE directives only.

If the old program library is sequential and F is not selected, called common decks that precede the decks that call them must be explicitly named on COMPILE directives to be audited. A common deck is audited automatically if it follows the deck that calls it. If the old program library is random, called common decks are audited automatically.

M MERGE PROGRAM LIBRARIES

This parameter merges two program libraries as one new program library. The M parameter is ignored on a creation run.

omitted No merge file.

M Program library to be merged with the old program library is on file MERGE.

M=lfn Program library to be merged with old program library on file named lfn.

N NEW PROGRAM LIBRARY FILE NAME

This parameter specifies the name of the new program library.

omitted Suppress new program library generation if correction run, otherwise write new program library to file named NEWPL.

N Write new program library to file named NEWPL.

N=lfn Write new program library to file named lfn.

O LISTABLE OUTPUT FILE NAME

This parameter specifies the name of the output file. Output file content is determined by the L parameter.

omitted or O Write output to file named OUTPUT.

O=lfn Write output to file named lfn.

P OLD PROGRAM LIBRARY FILE NAME

This parameter specifies the name of the old program library; it is ignored on a creation run.

omitted or P Old program library resides on file named OLDPL.

P=lfn Old program library resides on file named lfn.

P=lfn/s1/s2/ . . ./s7 Old program library resides on file named lfn. Secondary old program libraries reside on files s1, s2, . . . , s7.

P=/s1/s2/.../s7 — Old program library resides on file OLDPL. Secondary old program libraries reside on files s1, s2, . . . , ss7.

Q QUICK UPDATE MODE

This parameter specifies quick Update mode. It takes precedence when both F and Q are specified.

omitted — When F is also omitted, normal selective Update mode.

Q — Quick mode.

Corrections other than ADDFILE that reference cards in decks not specified on COMPILE directives are not processed in quick mode and Update abnormally terminates after printing the unprocessed corrections.

In Q mode, using a random old program library, a single correction set containing corrections to both a DECK and a COMDECK may cause trouble if the COMDECK logically precedes the DECK on the old program library. No errors will be detected, but if the same run is repeated with the N parameter specified on the UPDATE control statement and/or the old program library is sequential, the sequence numbers assigned to the text cards in the correction set will not be the same as they were in the Q mode run. This situation cannot be prevented without sacrificing the speed for which Q mode was designed. The correct sequence numbers are those assigned when N is specified or the old program library is sequential.

R REWIND FILES

This parameter specifies files to be rewound before and after an Update run.

omitted — Rewind the old program library, the new program libary, the compile file, the source file, and the pullmod file.

R — Do not rewind any files.

R=c. . .c — Each character in string indicates a file to be rewound.

- C Compile
- N New program library
- P Old program library and merge library
- S Source and pullmod

S SOURCE FILE NAME

This parameter specifies the name of the source file. The content of the source file is determined by the mode in which Update is operating, by the decks named on COMPILE directives, and by the format of the old program library in use (random or sequential).

omitted — Suppress source output file unless it is selected by the T parameter.

S — Source output file to be written on file named SOURCE.

S=lfn — Source output file to be written on file named lfn.

T OMIT COMMON DECKS FROM SOURCE FILE

This parameter specifies that common decks are to be excluded from the source file. It takes precedence over the S parameter.

omitted — Suppress source file unless it is selected by the S parameter.

T — Source output to be written on file named SOURCE, with common decks excluded.

T=lfn — Source output to be written on file named lfn, with common decks excluded.

U DEBUG HELP

The U parameter does not prevent Update from proceeding to pass 2 (correction phase) if errors are encountered in pass 1 (read-input-stream phase). The user should be aware that because of the method in which Update works, pass 1 errors could conceivably cause the flagging of pass 2 items which are not errors.

omitted — Update execution terminates when a fatal error is encountered.

U — Update execution is not terminated by a fatal error.

W SEQUENTIAL NEW PROGRAM LIBRARY FORMAT

This parameter specifies that the new program library is to have sequential format.

omitted — New program library format is determined by file residence as shown in table 2-3.

W — New program library is a sequential file.

X COMPRESSED COMPILE FILE

This parameter specifies that the compile file is to be compressed.

omitted — Compile file is not written in compressed format.

X — Compile file is written in compressed format (appendix D).

8 CARD IMAGE WIDTH ON COMPILE FILE

This parameter specifies total card image width on the compile file including seqeuncing information (appendix D).

omitted — Compile file output is composed of 90-column card images.

8 — Compile file output is composed of 80-column card images.

* MASTER CONTROL CHARACTER

This parameter specifies the master control character. If the character specified for a correction run is not the same as the character used when the old program library was created, the old program library character is used.

omitted — The first character of each directive is *.

*=c — The first character of each directive for this Update run is c; c can be any character A through Z, 0 through 9, or + − * / $ or =. (The $ character should be specified as *=$$$$ or /=$$$$.)

/ COMMENT CONTROL CHARACTER

This parameter specifies the comment control character.

omitted — Comment control character is /.

/=c — The comment control character is c; c can be any character A through Z, 0 through 9, or + − * / $ or =. (The $ character should be specifeid as *=$$$$ or /=$$$$.) Note, however, that the character should not be changed to one of the abbreviated forms of a directive unless NOABBREV is in effect.

UPDATE CONTROL CARD EXAMPLES

The UPDATE control statement

```
UPDATE(C=O,I=IN, L=F,N=TEST2,P=TEST1,S,*=+)
```

selects the following options in addition to default values for the omitted parameters:

C=O — A compile file is not generated.

I=IN — The input stream is on the file named IN.

L=F — A full output listing is generated.

N=TEST2 — A new program library named TEST2 is generated.

P=TEST1 — The old program library is on the file named TEST1.

S — A source file is generated on file named SOURCE.

*=+ — The master control character is +.

The UPDATE control statement

```
UPDATE(A,N=RAN,P=SEQ)
```

causes Update to copy the sequential old program library, SEQ, to a random new program library named RAN. The L, O, *, and / parameters assume their default values. No other parameters are applicable when A is specified.

The UPDATE control statement

```
UPDATE.
```

selects the following default values:

C=COMPILE

G=SOURCE (correction run)

I=INPUT

L=A,1,2 (creation run)
A,1,2,3,4 (correction run)

N=NEWPL (creation run)

O=OUTPUT

R=C,N,P,S

P=OLDPL (correction run)

=

/=/

In addition; the following defaults apply:

The compile file has 90 columns with 72 columns for data.

No editing is performed.

Update mode is normal selective.

The character set used is that specified in the library header.

No merging is performed.

Execution is terminated if a fatal error occurs .

New program library file format is determined by file residence.

The compile file is not in compressed format.

EXAMPLES OF UPDATE RUNS

This section contains several examples of Update runs. The directives illustrated include ADDFILE, PULLMOD, yanking and purging. Examples also show how to save a program library as a permanent file under the various operating systems. Also included in this section is an example of a FORTRAN Extended program maintained as a program library.

LIBRARY FILE CREATION

Figure 5-1 shows an example of an Update creation run in which several COMPASS and FORTRAN routines become a program library. The UPDATE control statement indicates a new library is to be created with the name PL. Since no other parameters are specified, Update uses their default values.

```
job statement
.
.
.
UPDATE(N=PL)
.
.
7/8/9
*DECK COMGROUP
  COMPASS program
*DECK COMGROUP1
  COMPASS program
*WEOR
*DECK FORGROUP
  FORTRAN program
*DECK FORGROUP2
  FORTRAN program
6/7/8/9
```

Figure 5-1. Update Creation Run

Since the first directive encountered is DECK, Update recognizes a creation run and begins construction of a new program library. All cards following the first DECK directive, up until the second DECK directive, are writen as a deck with the name COMGROUP. The first card is assigned the identifier COMGROUP.2, the next COMGROUP.3, and so forth. (The DECK directive itself is also a part of the library and has the identifier COMGROUP.1.)

A new deck, with card identifiers in the form COMGROUP1.n, begins when Update encounters the second DECK directive. In this example (figure 5-1), two COMPASS programs form the first two decks; COMGROUP and COMROUP1; two FORTRAN programs make up the last two decks: FORGROUP and FORGROUP1. At the end of the Update run, a program library exists with four decks.

The compile file produced by the run in figure 5-1 contains two system-logical records as a result of the WEOR directive. All four decks are written to the compile file. It has the default name of COMPILE.

The example in figure 5-2 shows a creation run in which directives are read from the alternate input file REMTAPE. Update reads text and directives from REMTAPE until the end of the system-lgoical record is encountered. Update then resumes reading from the main input file, INPUT. The resulting new program library contains decks A, B, C, and LOCAL.

```
A.  Update job deck.

    job statement
    .
    .
    .
    UPDATE(N)
    .
    .
    .
    7/8/9
    *READ REMTAPE
    *DECK LOCAL
      text of LOCAL
    6/7/8/9

B.  Contents of REMTAPE

    *DECK A
      text of A
    *DECK B
      text of B
    *DECK C
      text of C
```

Figure 5-2. Creation of Library From Alternate Input File

The program library, NEWPL, created by the example in figure 5-3 contains four decks, two of them are common decks. The compile file that is produced by default contains decks XA and XB in that order. Deck XB is expanded by Update to contain common deck D2 on the compile file.

```
job statement
.
.
.
UPDATE(N)
.
.
.
7/8/9
*COMDECK D1
  text of D1
*COMDECK D2
  text of D2
*DECK XA
  text of XA
*DECK XB
  text of XB
*CALL D2
6/7/8/9
```

Figure 5-3. Creation of Library With Common Decks

INPUT FILE NOT INPUT

Text and directives do not have to be part of the job deck. They can be in a file specified by the I parameter of the UPDATE control statement. In figure 5-4, Update creates a program library from information contained in file A1. The library that is produced contains three decks having cards identified by their deck name and sequence number as shown in figure 5-5.

```
A.  Update Job Deck
    job statement
    .
    .
    .
    UPDATE(I=A1, N)
    .
    .
    .
    6/7/8/9

B.  Contents of A1
    *COMDECK CSET
            COMMON A,B,C
    *DECK SET1
            PROGRAM ZIP
    C       A DO-NOTHING JOB
            STOP
            END
    *DECK SET2
            SUBROUTINE JIM
            A = B - SIN(C)
            RETURN
            END
```

Figure 5-4. Input File Not INPUT

```
*COMDECK CSET                          CSET.1
        COMMON A,B,C                   CSET.2
*DECK SET1                             SET1.1
        PROGRAM ZIP                    SET1.2
C       A DO-NOTHING JOB               SET1.3
        STOP                           SET1.4
        END                            SET1.5
*DECK SET2                             SET2.1
        SUBROUTINE JIM                 SET2.2
        A = B - SIN(C)                 SET2.3
        RETURN                         SET2.4
        END                            SET2.5
```

Figure 5-5. Program Library Contents

INSERTIONS/DELETIONS/COPYING

The Update run illustrated in figure 5-6 modifies the decks SET1 and SET2 of the program library created by the run in figure 5-4. As a result of the correction run, SET1 appears in the compile file as shown in figure 5-7.

Figure 5-8 shows the modification of an old program library named FN and the production of an assembly listing. The compile file that is read by COMPASS contains deck XA since that deck was modified by Update.

```
job statement
.
.
.
UPDATE(N,F)
.
.
.
7/8/9
*IDENT ADD1
*DELETE SET1.3, SET2.4
*CALL CSET
        B=1.0
        C=3.14159
        CALL JIM
*COPY SET1, SET1.5
*COPY SET2, SET2.2
*CALL CSET
*COPY SET2, SET2.3, SET2.5
6/7/8/9
```

Figure 5-6. Modify Old Program Library

```
PROGRAM ZIP                               SET1.2
COMMON A,B,C                              CSET.2
B=1.0                                     ADD1.2
C=3.14159                                 ADD1.3
CALL JIM                                  ADD1.4
STOP                                      ADD1.5
END                                       ADD1.6
SUBROUTINE JIM                            ADD1.7
COMMON A,B,C                              CSET.2
A = B - SIN(C)                            ADD1.9
RETURN                                    ADD1.10
END                                       ADD1.11
```

Figure 5-7. Compile File Contents

```
job statement
.
.
.
UPDATE(P=FN)
.
.
.
COMPASS(I=COMPILE)
.
.
.
7/8/9
*IDENT CS1
*INSERT XA.1
  Insertions
*DELETE XA.20, XA.23
6/7/8/9
```

Figure 5-8. Correction Run

PURGING AND YANKING

The purge directives differ from the yank directives in that yank operations are temporary. Cards yanked from the program library are temporarily deactivated. They can be reactivated by a subsequent yank of the yank directive that deactivated the card images.

In contrast, any change made to a program library through a purge directive is permanent. A reversal of a purge operation is possible only through the re-introduction of the cards into the library as if they had not previously existed.

The YANK directive in figure 5-9 becomes the first card on the new program library. The identifier for this card is NEGATE.1. The effects of the YANK can be nullified in future runs (and consequently the effects of the correction set GOTTOGO are restored) by specifying

```
     *IDENT RESTORE
     *DELETE NEGATE.1

or   *IDENT RESTORE
     *YANK NEGATE

or   *PURGE NEGATE
```

If the correction set NEGATE contained other corrections as well as the YANK, the YANK could be permanently removed by specifying

```
     *SELPURGE YANK$$$.NEGATE
```

or it could be temporarily removed by specifying

```
     *SELYANK YANK$$$.NEGATE
```

```
job statement
.
.
.
UPDATE(P=LIB,N=NEWLIB)
.
.
.
7/8/9
8IDENT NEGATE
*YANK GOTTOGO
6/7/8/9
```

Figure 5-9. Use of YANK

The Update run in figure 5-10 returns a program library to a previous level. The program library LIBAUG was modified periodically over a number of months. LIBAUG is the most recent (August) version of the program library. This run recreates a library modified only through May. The run purges all modifications made after May (beginning with JUNMOD1 in the directory).

```
job statement
.
.
.
UPDATE(N=LIBMAY,P=LIBAUG,C=0)
.
.
.
7/8/9
*PURGE JUNMOD1,*
6/7/8/9
```

Figure 5-10. Return to Previous Level

The run in figure 5-11 permanently removes deck BAD from the library. LIB is the most recent program library. NEWBAD is the new program library with BAD purged. *PURDECK BAD oeprates so that any cards having the identifier BAD but physically located outside of the deck BAD are not purged.

```
job statement
.
.
.
UPDATE(P=LIB,N=NEWBAD,C=0)
.
.
.
7/8/9
*PURDECK BAD
6/7/8/9
```

Figure 5-11. Use of PURDECK

As a means of comparing the effects of YANK, SELYANK, and YANKDECK, consider the following:

```
*YANK OLDMOD
```

This directive causes all effects of the correction set OLDMOD on the entire library to be nullified. Card images introduced by OLDMOD are deactivated; card images deactivated by OLDMOD are reactivated.

```
*SELYANK OLDDECK.OLDMOD
```

This directive accomplishes the same effect as the YANK directive above except its effect is limited to card images within the deck OLDDECK.

```
*YANKDECK OLDDECK
```

This directive affects all card images in OLDDECK, without regard to which correction set they belong.

The effects of the purge directives PURGE, SELPURGE, and PURDECK work the same as the YANK directives except the results are permanent.

SELECTIVE YANKING

The text stream in figure 5-12 illustrates the use of the DO and DONT directives. The deck ZOTS had contained cards introduced by the correction set DART; a later correction set contained a YANK directive that yanked correction set DART. The user wishes to nullify a portion of the YANK that affects the cards following ZOTS.19 through ZOTS.244; all other cards belonging to the correction set DART are to remain yanked. Inserting a DO at ZOTS.19 and a DONT at ZOTS.244 causes Update to rescind the yank while writing the deck ZOTS to the compile file.

```
*IDENT          REST
*INSERT         ZOTS.19
*DO             DART
*INSERT         ZOTS.244
*DONT           DART
```

Figure 5-12. Use of DO and DONT

SELECTIVE WRITING TO COMPILE FILE

During the correction phase Update processes the following directive:

```
     *DEFINE ABC
```

It is placed in the YANK$$$ deck. PROG2, a deck to be written on the compile file, contains the following sequence:

```
*DECK PROG2
   .
   .
   .
*IF DEF,ABC
   .
   .
   .
*ENDIF
```

Since ABC is defined, all active cards between the IF and ENDIF pair are written as if they are part of PROG2. Removing the DEFINE from the YANK$$$ deck would cause these text cards to be skipped.

The input stream in figure 5-13 has mutually exclusive requirements depending on the availability of correction set IDC. If IDC is known, the first 15 active cards after the first IF are written to the compile file. If IDC is not known, the cards following the second IF through the ENDIF are written to the compile file.

```
*DECK DECKA
.
.
.
*IF IDENT,IDC,15
*IF - IDENT,IDC
 active text cards
*ENDIF
```

Figure 5-13. Use of IF and ENDIF

Nesting of IF directives is illustrated in figure 5-14. The deck ROCK has an IF-controlled sequence containing a second IF-controlled sequence. The text following the first IF is written if PEBBLE is known; the text following the second IF is written if both PEBBLE and STONE are known. The ENDIF terminates both IF controlled sequences.

```
*DECK ROCK
.
.
.
*IF IDENT,PEBBLE
.
.
.
*IF IDENT,STONE
.
.
.
*ENDIF
```

Figure 5-14. Nexted IF Directives

ADDITION OF DECKS

A new program library, NEWPL, is to be constructed from the old program library, OLDPL, with the addition of one new common deck and two new decks. The new common deck, D1A, will be the first deck after the YANK$$$ deck; the new deck XC will follow deck SX; and the new deck SYSTEXT will be the last deck on the new program library. No compile file will be produced. All three of the ADDFILEs in figure 5-15 are to be read from the main input file INPUT. The ADDFILEs in figure 5-16 are to be read from the main input file INPUT. The ADDFILEs in figure 5-16 are to be read from the Update input file FNAME. In both these cases, the input file need not be specified but the two separators must be included (either space and comma or two commas). Each of the ADDFILE directives in figure 5-17 will cause Update to read from a separate file that is not the main input file. Common deck D1A and its text are on FILEA; deck SYSTEXT and its text are on FILEB; deck XC and its text are on FILEC.

```
job statement
.
.
.
UPDATE(N,C=0)
.
.
.
7/8/9
*ADDFILE INPUT, YANK$$$ or *ADDFILE,, YANK$$$
*COMDECK D1A
.
.
.
*ADDFILE INPUT or *ADDFILE
*DECK SYSTEXT
.
.
.
*ADDFILE INPUT,XB or *ADDFILE,,XB
*DECK XC
.
.
.
6/7/8/9
```

Figure 5-15. ADDFILE Input on File INPUT

```
A.  Update run
    job statement
    .
    .
    .
    UPDATE(N,C=0,I=FNAME)
    .
    .
    .
    6/7/8/9

B.  Contents of file FNAME
    *ADDFILE FNAME, YANK$$$ or *ADDFILE,,YANK$$$
    *COMDECK D1A
    .
    .
    .
    *ADDFILE FNAME or *ADDFILE
    *DECK SYSTEXT
    .
    .
    .
    *ADDFILE FNAME,XB or *ADDFILE,,XB
    *DECK XC
    .
    .
    .
```

Figure 5-16. ADDFILE Input on File FNAME

```
job statement
.
.
.
UPDATE(N,C=0)
.
.
.
7/8/9
*ADDFILE FILEA,YANK$$$
*ADDFILE FILEB
*ADDFILE FILEC,XB
6/7/8/9
```

Figure 5-17. ADDFILE Input on Secondary Input Files

PULLMOD OPTION

The program library created by the example in figure 5-4 (Input File Not Input) has been altered by the correction run in figure 5-18. As a consequence of the run, the deck SET1 contains the following cards:

```
*DECK  SET1
       PROGRAM ZIP
C      THIS IS FOR PULLMOD EXAMPLE
       STOP
       END
```

```
job statement
.
.
.
UPDATE (N=PL2)
.
.
.
7/8/9
*IDENT PMEX
*DELETE SET1.3
C          THIS IS FOR PULLMOD EXAMPLE
*COMPILE SET1
6/7/8/9
```

Figure 5-18. Correction Run for PULLMOD Example

The Update run in figure 5-19 re-creates the correction set that changed SET1; the file PMFILE contains the following re-created correction set:

```
*IDENT PMEX
*DELETE SET1.3,SET1.3
C       THIS IS FOR PULLMOD EXAMPLE
```

```
job statement
.
.
.
UPDATE(G=PMFILE, P=PL2)
7/8/9
*PULLMOD PMEX
6/7/8/9
```

Figure 5-19. Pull Modifications

PROGRAM LIBRARY AS A PERMANENT FILE

The job deck in figure 5-20 illustrates the creation and saving of a program library as a permanent file under NOS/BE and SCOPE 2; the deck in figure 5-21 saves a program library as an indirect access file under NOS. See the appropriate operating system reference manual for additional details.

```
job statement
accounting statements
REQUEST(PL,*PF)
UPDATE(N=PL,W,L=1234)
CATALOG(PL,PLIB,ID=JONES)
7/8/9
*DECK ONE
.
.
.
6/7/8/9
```

Figure 5-20. Permanent File Under NOS/BE or SCOPE 2

```
job statement
accounting statements
UPDATE(N=PL,W,L=1234)
SAVE(PL=UPLIB).
7/8/9
*DECK ONE
.
.
.
6/7/8/9
```

Figure 5-21. Permanent File Under NOS

SAMPLE FORTRAN EXTENDED PROGRAM

This set of Update examples illustrates how Update can be used for maintaining a FORTRAN Extended program in program library format. The FORTRAN program is simple. It calculates the area of a triangle from the base and height read from the data record.

The job in figure 5-22 places the FORTRAN program and subroutine as a single deck (ONE) on the new program library (NEWPL) and on the compile file (COMPILE). Following Update execution, FTN is called to compile the program; the source is on the COMPILE file. LGO calls for execution of the compiled program. This program does not execute because of an error in the SUBROUTINE statement. The name of the subroutine should be MSG, not MSA.

```
job statement
.
.
.
UPDATE(N,F)
FTN(I=COMPILE)
LGO.
.
.
.
7/8/9
*DECK ONE
        PROGRAM ONE(INPUT,OUTPUT,TAPE1)
        PRINT 5
5       FORMAT(1H1)
10      READ 100, BASE,HEIGHT,I
100     FORMAT (2F10.2, I1)
        IF (I.GT.0) GO TO 120
        IF (BASE.LE.0) GO TO 105
        IF (HEIGHT.LE.0) GO TO 105
        GO TO 106
105     CALL MSG
106     AREA = .5 * BASE * HEIGHT
        PRINT 110, BASE, HEIGHT, AREA
110     FORMAT (///, * BASE=*F20.5, *  HEIGHT=*
       I F18.5, *  AREA=*F20.5)
        WRITE (1) AREA
        GO TO 10
120     STOP
        END
        SUBROUTINE MSA
        PRINT 400
400     FORMAT (///,* FOLLOWING INPUT DATA
       I NEGATIVE OR ZERO *)
        RETURN
        END
7/8/9
data
6/7/8/9
```

Figure 5-22. FORTRAN Extended Program Library – 1

Examination of Update output from the creation job reveals that the erroneous SUBROUTINE statement has card identifier ONE.20. The job in figure 5-23 corrects the error and generates a new program library.

```
job statement
.
.
.
UPDATE(N,F)
FTN(I=COMPILE)
LGO.
.
.
.
7/8/9
*IDENT MOD1
*DELETE ONE.20
        SUBROUTINE MSG
7/8/9
data
6/7/8/9
```

Figure 5-23. Correction of SUBROUTINE Statement

The job in figure 5-24 uses the same input as the job in figure 5-22 but divides the program into two decks, ONE and MSG. Deck MSG is a common deck. A CALL directive is inserted into deck ONE to assure that MSG is written on the compile file whenever deck ONE is.

```
job statement
.
.
.
UPDATE(N,F)
FTN(I=COMPILE)
LGO.
.
.
.
7/8/9
*COMDECK MSG
        SUBROUTINE MSG.
        PRINT 400
400     FORMAT (///,* FOLLOWING INPUT DATA
      I  NEGATIVE OR ZERO *)
        RETURN
        END
*DECK ONE
        PROGRAM ONE(INPUT,OUTPUT,TAPE1)
        PRINT 5
5       FORMAT(1H1)
10      READ 100, BASE,HEIGHT,I
100     FORMAT (2F10.2, I1)
        IF (I.GT.0) GO TO 120
        IF (BASE.LE.0) GO TO 105
        IF (HEIGHT.LE.0) GO TO 105
        GO TO 106
105     CALL MSG
106     AREA = .5 * BASE * HEIGHT
        PRINT 110, BASE, HEIGHT, AREA
110     FORMAT (///, * BASE=*F20.5, *  HEIGHT=*
      I  F18.5, *  AREA=*F20.5)
        WRITE (1) AREA
        GO TO 10
120     STOP
        END
7/8/9
data
6/7/8/9
```

Figure 5-24. FORTRAN Extended Program Library – 2

The example in figure 5-25 adds a deck to the library created in the previous example (figure 5-24). Since no new program library is generated (N is omitted from Update call), the addition is temporary.

```
job statement
.
.
.
UPDATE.
FTN(I=COMPILE)
LGO.
.
.
.
7/8/9
*IDENT MOD2
*INSERT ONE.20
*DECK TWO
        PROGRAM TWO(INPUT,OUTPUT)
          .
          .
          .
        END
*CALL MSG
*DELETE MSG.3
400     FORMAT(///, * FOLLOWING INPUT DATA
      I  POSITIVE*)
7/8/9
data
6/7/8/9
```

Figure 5-25. Add Deck to FORTRAN Program Library

STANDARD CHARACTER SET

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect throughout the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1; 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00[†]	: (colon)[††]	8-2	00	: (colon)[††]	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	%[††]	8-6	16	%[††]	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	↱	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2[†††]	52	!	12-8-7 or 11-0[†††]	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2[†††]	72	<	12-8-4 or 12-0[†††]	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

[†] Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

[††] In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

[†††] The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

DIAGNOSTICS B

Diagnostic messages can either appear in the dayfile or are intermixed with Update output in the output file. In addition to detecting errors, Update detects overlapping corrections when the EXTOVLP installation option has been assembled.

DIAGNOSTIC MESSAGES

All diagnostic messages that can be issued during an Update run are listed in alphabetic order in table B-1. One of the following codes is included for each diagnostic:

Type	Meaning
I	An informative message; processing continues.
N	A non-fatal error; processing continues.
F	A fatal error; processing is terminated.

OVERLAPPING CORRECTIONS

Update can detect four overlapping correction situations. When any of these types are detected, Update prints the offending line with the words TP.n OVLP appended on the far right. Type n is one of the following:

Type	Meaning
1	Two or more modifications are made to one card by a single correction set.
2	A modification attempts to activate an already active card.
3	A modification attempts to deactivate an already inactive card.
4	A card is inserted after a card which is inactive on the old program library and is inactive on the new program library.

The listing of overlap lines is controlled by list option 3.

Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory, and they point to conditions in which the probability of error is greater than normal. If any overlap condition is encountered, a dayfile message is printed.

Type TP.2 and TP.3 are detected by comparing existing correction history bytes with those to be added. Complex operations involving YANK and PURGE might generate these overlap messages even though no overlap occurs.

TABLE B-1. DIAGNOSTICS

Message	Type	Significance	Action
A OPTION INVALID WITH RANDOM OLDPL OR SEQUENTIAL NEWPL	F	The old program library is not sequential or the new program library is not random or is not on a random device for a sequential-to-random copy.	Correct the error.
ADDFILE CARD INVALID ON REMOTE FILE	F	The ADDFILE directive cannot be used in the file specified by a READ directive.	Remove the ADDFILE directive from the file specified by the READ directive.
ADDFILE FIRST CARD MUST BE DECK OR COMDECK	F	The first card on the file specified by the ADDFILE directive is not a DECK or COMDECK directive.	Correct the error.
ALL YANK, SELYANK, YANKDECK, AND CALL CARDS AFFECTED HAVE BEEN CHANGED	I	If Update changes any identifiers during a merge, it also changes the corresponding YANK, SELYANK, YANKDECK, and CALL directives.	None.
B OPTION INVALID WITH SEQUENTIAL OLDPL	E	The old program library is not random for a random-to-sequential copy.	Do not specify B on the control statement.
BAD ORDER ON YANK DIRECTIVE	N	Identifiers separated by a period on the YANK directive are in the wrong order.	Correct the order of the identifiers.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
CARD NUMBER ZERO OR INVALID CHARACTER IN NUMERIC FIELD	F	Sequence number field on the correction directive is erroneous.	Correct the sequence number.
***CONTROL CARD INVALID OR MISSING	F	Update detected a format error on a directive, deleted, a directive that was unrecognizable, or detected an illegal file name. Illegal operations such as INSERT prior to an IDENT could also have been attempted.	Correct the error.
COPY TO EXTERNAL FILE NOT ALLOWED WHEN REAOING ATERNATE INPUT UNIT	N	No copy is made.	Correct the error.
COPYING INPUT TO TEMPORARY NEWPL	1	A sequential new program library was requested on a creation run.	None.
COPYING OLOPL TO A RANOOM FILE	1	The old program library is being copied to a random file.	None.
CREATING NEW PROGRAM LIBRARY	I	Indicates that a new program library is being created.	None.
DECK NAME ON ABOVE CARO NOT LAST DECLARED DECK	1	When a DECLARE directive is in effect, only card images belonging to decks specified can be modified or referenced.	Add appropriate OECLARE directives or remove directives which reference non-declared decks.
OECK SPECIFIED ON MOVE OR COPY CARD NOT ON OLDPL, CARO WILL BE IGNORED	I	The specified deck will not be moved or copied.	Correct the error.
DECK STRUCTURE CHANGED	I	A deck has been moved or deleted.	None.
DO/OONT IOENT idname IS NOT YANKED/YANKEO NULL OO/OONT	1	A OO directive to negate the effect of a YANK references an identifier that has been yanked, or a DONT directive to restore a YANK references on identifier that was already yanked.	None.
OUPLICATE OECK dname NEWPL ILLEGAL	F/N	Update encountered an active OECK of COMOECK directives that duplicates a previous directive. This condtion is fatal if a new program library is being created; nonfatal is a new program library is not being created.	Change one of the deck names.
DUPLICATE FILE NAME OF file, JOB ABORTEO	F	Same file name has been assigned to two Update files.	Change one of the file names.
OUPLICATE IOENT CHANGED TO ident	N	Update changed a duplicate identifier name to a unique one.	None.
OUPLICATE IDENT NAME	F	During a merge run, Update encountered a duplicate identifier name that it coult not make unique.	Change one of the identifiers.
OUPLICATE IDENT NAME IN AOOFILE	F	The name of a correction set to be added as a result of an AOO-FILE directive duplicates a correction set name on the old program library.	Change the name of the correction set.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
DUPLICATE SECONDARY OLDPL IGNORED	I	Two secondary old program libraries have the same name.	Correct the error or ignore.
ERRORNOT ALL MODS WERE PROCESSED***	F	All changes indicated in the input stream were not processed.	Make sure that names specified on correction directives correspond to identifiers on the old program library (or on the COMPILE directive if in quick mode).
FILENAME OF file IS TOO LONG, UPDATE ABORTED	F	A file name exceeds seven characters.	Correct the file name.
FILENAME ON ABOVE CARD GREATER THAN SEVEN CHARACTERS	F	A file name exceeds seven characters.	Correct the file name.
FILE NAME ON UPDATE CARD GR 7 CHARACTERS	F	A file name on the UPDATE control statement is greater than seven characters.	Correct the error.
G AND O FILES CANNOT HAVE SAME FILENAME	F	The G and O control statement options specify the same file name.	Change one of the names.
GARBAGE IN OLDPL HEADER, UPDATE ABORTED	F	Invalid data was found in the random index.	Rerun job/recreate program library. If the problem still exists, notify systems analyst.
IDENT CARD MISSING, NO NEWPL REQUESTED, DEFAULT IDENTIFIER OF .NO.ID. USED	I/F	If no new program library is generated, then a correction set need not be introduced by an IDENT directive. The identifier .NO.ID. is used.	Add IDENT directive if new program library is to be generated.
IDENT LONGER THAN NINE CHARACTERS	F	An identifier can only have up to nine characters.	Correct the identifier.
IDENTIFIERS SEPARATED BY PERIOD IN WRONG ORDER	F	The specified identifiers are not the correct order.	Switch the identifiers.
ILLEGAL CONTROL CARD IN ADDFILE	F	ADDFILE insertions cannot contain correction directives.	Remove the correction directives.
IMPROPER MASTER CHARACTER CHANGED TO char	N	The character specified on the * control statement parameter is not the same as the master control character on the old program library.	Use the same master control character as on the old program library.
INSUFFICIENT FIELD LENGTH, UPDATE ABORT	F	The table manager ran out of room for internal tables.	Allocate more field length.
IT MAY EXIST IN A DECK NOT MENTIONED ON A COMPILE CARD	F	An identifier references a card in a deck not specified on a COMPILE directive (only if in quick mode).	Correct the error.
INVALID NUMERIC FIELD	F	The directive does not contain required numeric field.	Correct the directive.
LENGTH ERROR ON OLDPL. UNUSABLE OLDPL OR HARDWARE ERROR	F	Card length on old program library is greater than the maximum allowed or is less than one.	Rerun job. If problem still exists, then recreate the program library.
LISTED BELOW ARE ALL IDENT NAMES WHICH WERE CHANGED DURING THE MERGE	I	Update changes any duplicate identifiers to make them unique when merging two program libraries.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
NEW IDENT ON CHANGE CARD IS ALREADY KNOWN	F	An attempt was made to change a correction set identifier to one already in existence.	Correct the error.
NO ACTIVE CARDS WERE FOUND WITHIN THE COPY RANGE. NULL COPY	N	All card images within the specified range are inactive.	None.
NO DECK NAME ON DECK CARD	F	No name was specified on the DECK directive.	Specify a name.
NO INPUT FILE, Q MODE, UPDATE ABORT	F	In quick mode, Update relies on the input file to determine what is written to the compile file.	Put appropriate COMPILE directives in the input file.
NO OLDPL, NOT CREATION RUN, UPDATE ABORT	F	No old program library was supplied on a non-creation run.	Correct the error.
NULL ADDFILE	I	The first read on the file specified by ADDFILE encountered an end-of-record. If the input file was specified, the first read encountered an illegal directive.	Correct the error.
NULL IDENT	F	An identifier was not found on a directive where one was expected.	Correct the directive.
NULL DECK NAME	F	During ADDFILE or a CREATION run, Update encountered a DECK or COMDECK directive that did not have a name.	Correct the directive.
OLDPL READ ERROR - POSSIBLE LOST DATA AFTER FOLLOWING CARD card image ***AND BEFORE THE FOLLOWING CARD*** card image	F	A parity error on other error has occurred while processing an old program library. As a result Update is uncertain of the position of the old program library. The first card shown is the last card Update successfully processed. The second card is the next valid card that Update was able to find following the error.	Rerun the job.
OLDPLS HAVE DIFFERENT CHARACTERS SETS	I	The merging of two old program libraries with different character sets is not allowed.	Use program libraries with the same character set.
OUTPUT LINE LIMIT EXCEEDED. LIST OPTIONS 3 AND 4 DEFEATED	N	Update output exceeds the line limit specified by default or by the LIMIT directive.	Use the LIMIT directive to increase one unit.
PLS HAVE DIFFERENT CONTROL CHARACTERS, ABORT	F	The merging of two program libraries with different control characters is not allowed.	Use program libraries with the same control characters.
PREMATURE END OF RECORD ON OLD PROGRAM LIBRARY	F	A PRU of level 0 was encountered in the midst of a card image.	Rerun the job. If error still exists, recreate the program library.
RANDOM NEWPL CANNOT BE A BLOCKED FILE	F	The new program library cannot be blocked (RT=S) when the A parameter is specified on the UPDATE control statement under SCOPE 2.	Unblock the new program library.
READING INPUT	I	The input file is being read by Update.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
RECURSIVE CALL ON COMDECK dname IGNORED. FATAL ERROR	F	A common deck has called itself or common decks that contain calls to the specified common deck.	Correct the error.
SECONDARY OLDPL NOT RANDOM	F	Secondary old program libraries must be random.	Use random secondary old program libraries.
SEQUENCE NUMBER EXCEEDS 131071	F	The proper range of sequence numbers is 1 thorugh 131071.	Correct the error.
STACK DEPTH EXCEEDED	F	Stack in which card images are placed became full while processing a BEFORE or ADDFILE directive.	Notify systems analyst (increase RECURDEP).
TABLE MANAGER LOGIC ERROR	F	There is not enough table space to accommodate the old program library tables.	Increase field length.
THE ABOVE CALLED COMMON DECK WAS NOT FOUND	F	The called common deck could not be found.	Check the spelling of the deck name. If creating a program library with calls to secondary dd program libraries, set C=0 on the UPDATE control statements.
THE ABOVE CARD IS ILLEGAL DURING A CREATION RUN	F	A directive that is not allowed on a creation run was encountered.	Remove the illegal directive.
THE ABOVE CONTROL CARD IS ILLEGAL AFTER A DECK HAS BEEN DECLARED	N	CHANGE, PURGE, and YANK directives are illegal after a deck has been specified on a DECLARE directive. They are ignored.	Remove the illegal directives.
THE ABOVE LISTED CARDS CANNOT EXIST IN THE YANK DECK AND HAVE BEEN PRUGED DURING EDITING	I	Only YANK, YANKDECK, SELYANK, and DEFINE directives are kept in the YANK$$$ deck.	None.
THE ABOVE OPERATION IS NOT LEGAL WHEN REFERENCING THE YANK DECK	F	The specified operation is illegal when referencing the YANK$$$ deck.	Correct the error.
THE ABOVE SPECIFIED CARD WAS NOT ENCOUNTERED	F	Update could not locate the specified card on the old program library.	Make sure that the correct identifier is specified.
THE INITIAL CARD OF THE COPY RANGE WAS NOT FOUND. NULL COPY	N	No copy was made.	Make sure that the correct identifier is specified.
THE TERMINAL CARD OF THE COPY RANGE WAS NOT FOUND. COPY ENDS AT END OF SPECIFIED DECK	I	The terminal card specified was not found - the rest of the deck was copied.	Make sure that the correct identifier is specified.
THE TERMINAL CARD SPECIFIED WAS NOT ENCOUNTERED	F	While processing a card range, Update could not locate the last card of the range.	Make sure that the correct identifier is specified.
THIS UPDATE REQUIRED n WORDS OF CORE	I	It took n words of memory for the update.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance	Action
TOO MANY CHBS -- INCREASE L.CHB	F	Correction history bytes exceed the specified limit of 100_8 for a card.	Notify systems analyst.
TOO MANY SECONDARY OLDPLS SPECIFIED	F	Up to seven secondary old program libraries can be specified.	Specify seven or less secondary old program libraries.
UNBALANCED TEXT/ENDTEXT CARDS, LAST ENDTEXT CARD IGNORED	N	Update encountered more ENDTEXT directives than TEXT directives.	None.
UNKNOWN IDENTIFIER idname	F	A correction directive references an identifier not found in the directory.	Make sure that the correct identifier is specified.
UPDATE COMPLETE	I	The update is completed.	None.
UPDATE CONTROL CARD ERROR(S)	F	The UPDATE control statement contains unacceptable parameters. The erroneous parameters are listed on the next line.	Correct the erroneous parameters.
UPDATE CREATION RUN	I	This Update run was a creation run.	None.
WAITING FOR 45000B WORDS	I	Update is waiting for the operating system to allocate it enough memory.	None.
WARNINGOLDPL CHECKSUM ERROR***	I	At least one updated deck from the old program library is bad.	Rerun job. If problem still exists, notify systems analyst.
WARNING, RETURNING PRIOR NEWPL	I	Two consecutive Updates were processed, each of which created a random new program library of the same name. Update returns the new program library created by the first update.	To save both new program libraries, give them unique names.
YANK, SELYANK, OR YANKDECK ident NOT KNOWN	N	The identifier referenced on a YANK, SELYANK, or YANKDECK has probably been purged; this applies to cards already on the library.	Remove the yank directive from the YANK$$$ deck.
deckname IS NOT A VALID DECK NAME	F	A deck name has 1 through 9 characters; legal characters are: A through Z, 0 through 9, and + - * / ( ) $ =.	Correct the deck name.
n ERRORS IN INPUT	I	Update encountered n fatal errors in the input stream. Processing continues in order to detect additional errors. This message is issued only if the U parameter is specified on the control statement.	None.
n ERRORS IN INPUT, NEWPL, COMPILE, SOURCE SUPPRESSED	I	Update encountered n fatal errors in the input stream. Processing continues in order to detect errors. A new program library, a compile file, and a source file is not generated.	None.

TABLE B-1. DIAGNOSTICS (Contd)

Message	Type	Significance		Action
n ERRORS IN UPDATE INPUT	I	First pass of Update processing encountered n fatal errors while reading a correction set.	None.	
n DECLARE ERRORS	I	Indicates the number of directives that reference card images in decks not specified on DECLARE directives.	None.	
n FATAL ERRORS	I	Indicates the number of errors which caused Update to abort.	None.	
n NONFATAL ERRORS	I	Indicates the number of errors which did not cause Update to abort.	None.	
n OVERLAPPING CORRECTIONS	I	A correction set changed the status of some cards more than once or referenced an inactive card image.	None.	
n UPDATE ERRORS, JOB ABORTED	I	Errors were encountered in reading the input file.	None.	

GLOSSARY

CARD IDENTIFIER – The combination of identifier and sequence number that uniquely identifies each card image in a program library.

COMMON DECK – A deck that is written on a compile file as a result of a CALL directive. The COMDECK directive introduces a common deck.

COMPILE FILE – The file generated by Update that contains card images restored to a format that is acceptable to a compiler or assembler.

COPY RUN – An Update run that performs a sequential-to-random or random-to-sequential copy of a program library. Contrast with creation run and correction run.

CORRECTION HISTORY BYTE – A byte added to a card image by Update each time the status of the card image changes. The correction history byte tells Update whether or not a card image is active or inactive and which correction set modified the card image.

CORRECTION RUN – An Update run in which changes can be made to a program library. Contrast with copy run and creation run.

CORRECTION SET – A set of directives and text that direct Update to modify a program library. The IDENT directive introduces a correction set.

CREATION RUN – An Update run that constructs a program library. It is the original transfer of cards into Update format. Contrast with copy run and correction run.

DECK – A deck consists of a DECK or COMDECK directive and all text and directives until the next DECK or COMDECK directive. It is the smallest unit that can be extracted from a program library.

DECK LIST – A list internal to Update that contains the names of all decks in the program library and the location of the first word for each deck.

DIRECTORY – A list that contains one entry for each DECK, COMDECK, and IDENT directive that is used for the program library.

FULL UPDATE MODE – An Update run in which the F parameter is selected on the control statement causing Update to process all decks on the library. Contrast with normal selective mode and quick Update mode.

IDENTIFIER – The name of a deck, common deck, or correction set.

INPUT FILE – The user-supplied file or part of the job deck that contains the input stream of Update directives and text.

MASTER CONTROL CHARACTER – A character in column 1 that informs Update that the card contains a directive.

MERGE FILE – The file that contains a program library to be merged with the old program library into a new program library.

NEW PROGRAM LIBRARY – The program library initially generated on a creation run. A new program library that incorporates the chagnes made during a correction run is generated if requested.

NORMAL SELECTIVE MODE – An Update run in which the F and Q options are not selected on the control statement. All decks specified on COMPILE directives as well as all corrected decks are processed. Contrast with full Update mode and quick Update mode.

OLDPROGRAM LIBRARY – The program library to be modified.

OUTPUT FILE – The file generated by Update that contains the status information produced during Update execution. It is in a form suitable for printing.

PROGRAM LIBRARY – The file generated by an Update run that contains the decks of card images. Card images in the program library are in a format that can be manipulated by Update, but that is meaningless for all other purposes.

PULLMOD FILE – A file that contains directives and text or recreated correction sets specified on PULLMOD directives.

QUICK UPDATE MODE – An Update run in which the Q option is selected on the control statement. Only decks specified on COMPILE directives and called common decks are processed. Contrast with full Update mode and normal selective mode.

SECONDARY OLD PROGRAM LIBRARY – A program library from which decks on the old program library can call common decks.

SEQUENCE NUMBER – A number supplied by Update that uniquely identifies a card image.

SOURCE FILE – The file generated by Update that contains card images of an input stream that would allow regeneration of the program library.

SYSTEM-LOGICAL RECORD – Under NOS/BE, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure.

Equivalent to a logical record under NOS.

The following table shows equivalency under SCOPE 2.

Type	Level	Equivalency
RT=W	0 thru 16_8	end-of-section
RT=W	17_8	end-of-partition
RT=S	0 thru 17_8	end-of-record
RT=Z	0 thru 17_8	end-of-section
BT=C	0 thru 17_8	end-of-section

FILE FORMAT AND STRUCTURE D

The files generated and used by Update have formats determined by both the operating system in use and the user. This appendix describes default file formats, allowed file formats, and the interchangeability of files among operating systems. Table D-1 summarizes file structure according to the operating system used.

LIBRARY FILE FORMATS

Update can create and maintain library files in two distinctly different formats: random and sequential. These formats are described in detail below. Random format should be used whenever possible because it can be processed substantially faster than sequential format.

RANDOM FORMAT

On a random format library, each deck is a system-logical-record as shown in figure D-1. The deck records are followed by records containing the deck list, the directory, and the random index.

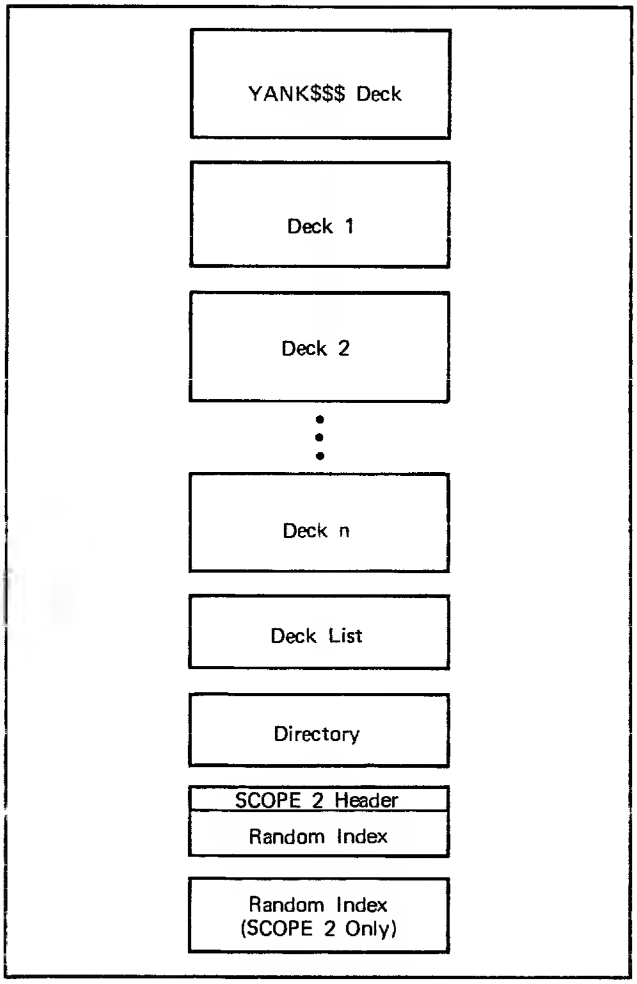


Figure D-1. Random Program Library Format

Random Index

The random index tells Update where the directory and deck list begin and how long they are. The index also contains such information as what master control character and which character set was used when the library was generated. Random index format is shown in figure D-2.

Two copies of the random index are generated under SCOPE 2 because Update generates another copy when it closes the file. The closing of the file is a process internal to Update.

Under SCOPE 2, Update adds a two word header to the random index that indicates the number of words in the index. SCOPE 2 header format is shown in figure D-3.

Copying to Tape

Random program libraries should be copied to tape through Update parameters. To copy a random program library to tape under NOS or NOS/BE, use the UPDATE control statement

```
UPDATE(B,P=plname,N=lfn)
```

where plname is the library name and lfn is the tape file. To copy the library back to mass storage, use

```
UPDATE(A,P=lfn,N=newpl)
```

where lfn is the tape file and newpl is the new program library name.

Under SCOPE 2, use the UPDATE control statement

```
UPDATE(F,P=plname,N=lfn)
```

to copy a random program library to tape. The program library name is plname and lfn is the tape file. To copy the library back to mass storage, use

```
UPDATE(F,P=lfn,N=newpl)
```

where lfn is the tape file and newpl is the new program library name.

SEQUENTIAL FORMAT

Update optionally creates new program libraries in sequential format. On magnetic tape, a sequential library (SI tape format) is written as one record in binary (figure D-4). The first word in the file is a display code key word (figure D-5); the second is a counter word containing the number of deck names in the deck list and the count of correction set identifiers in the directory (figure D-6). The last word in the file is a checksum (figure D-7).

TABLE D-1. FILE STRUCTURE AND OPERATING SYSTEM

Update Files	NOS/BE		NOS		SCOPE	
	Tape	Mass Storage	Tape	Mass Storage	Tape	Mass Storage
P=OLDPL	Binary	Random or sequential	Binary	Random or sequential	Binary, sequential[†] RT=W or S	Random: RT=W unblocked Sequential: RT=W unblocked RT=W
N=NEWPL	Binary	Random or W - sequential	Binary	Random W - sequential	Binary, sequential RT=W or S	Random if unblocked Sequential if blocked or if W specified on Update control statement. RT=W unblocked by default. RT=blocked W or S; specified through FILE control statement. Cannot be blocked if random.
C=COMPILE	NOS/BE coded	Sequential	Determined by REQUEST control statement	Sequential	RT=W, I blocked. Other types determined by FILE control statement	RT=W unblocked RT=S if compressed file; S specified through FILE control statement.
I=INPUT	NOS/BE coded	Sequential	Determined by REQUEST control statement	Sequential	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE control statement.	RT=W unblocked RT=W blocked or RT=Z, FL≤100 through FILE control statement
O=OUTPUT	NOS/BE coded	Sequential	Determined by REQUEST control statement	Sequential	RT=W, I blocked. Other types possible through FILE control statement.	RT=W, unblocked
S=SOURCE	NOS/BE coded	Sequential	Determined by REQUEST control statement	Sequential	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE control statement.	RT=W, unblocked RT=W blocked or RT=Z if specified through FILE control statement.
*READ	NOS/BE coded	Sequential	Determined by REQUEST control statement	Sequential	RT=W, I blocked. Other blocking or RT=Z, FL≤100 through FILE control statement.	RT=W, unblocked RT=W blocked or RT=Z if specified through FILE control statement.

[†] Random files can be put on tape by copying the file to tape. To access this file, it must first be copied to a W unblocked file. W records are 5120 characters in length. SCOPE 2 Update checks for presence of directory header containing DIRECT$ to identify random file and for presence of CHECK in word 1 of sequential file. If both tests fail, library format is unacceptable. Random format library must be unblocked W records.

<table>
<tr><th>59</th><th>47</th><th>29</th><th></th><th>23</th><th>17</th><th>11</th><th>5 0</th></tr>
<tr><td>7000</td><td>dll</td><td colspan="6">dllra</td></tr>
<tr><td>unused</td><td>dirl</td><td colspan="6">dirra</td></tr>
<tr><td colspan="3">unused</td><td>m</td><td>x</td><td>lab</td><td>y</td><td>c</td></tr>
<tr><td colspan="8">label</td></tr>
<tr><td colspan="8">label (cont.)</td></tr>
</table>

7000 Identifies random directory record.

dll Length of the deck list in words.

dllra Random address of first word of deck list.

dirl Length of directory in words.

dirra Random address of first word of directory.

m Indicates presence of deck bits in deck list

1	deck bits present
other	deck bits not present

x Character set identifier determined by IP.CSET parameter.

3 (36_8)	IP.CSET is set for a 63-character set
4 (37_8)	IP.CSET is set for a 64-character set

lab Label flag:

nonzero	words 3 and 4 contain tape label.
0	words 3 and 4 not present

SCOPE 2 does not recognize tape labels.

y Indicates which character set was used when the library was generated.

Y or null	64 character set used
other	63 character set used

c Indicates master control character in use when the library was created.

Figure D-2. Random Index Format

<table>
<tr><th>59</th><th>17 0</th></tr>
<tr><td>DIRECT$</td><td>unused</td></tr>
<tr><td colspan="2">n</td></tr>
</table>

n Number of words in the random index.

Figure D-3. SCOPE 2 Random Index Header Format

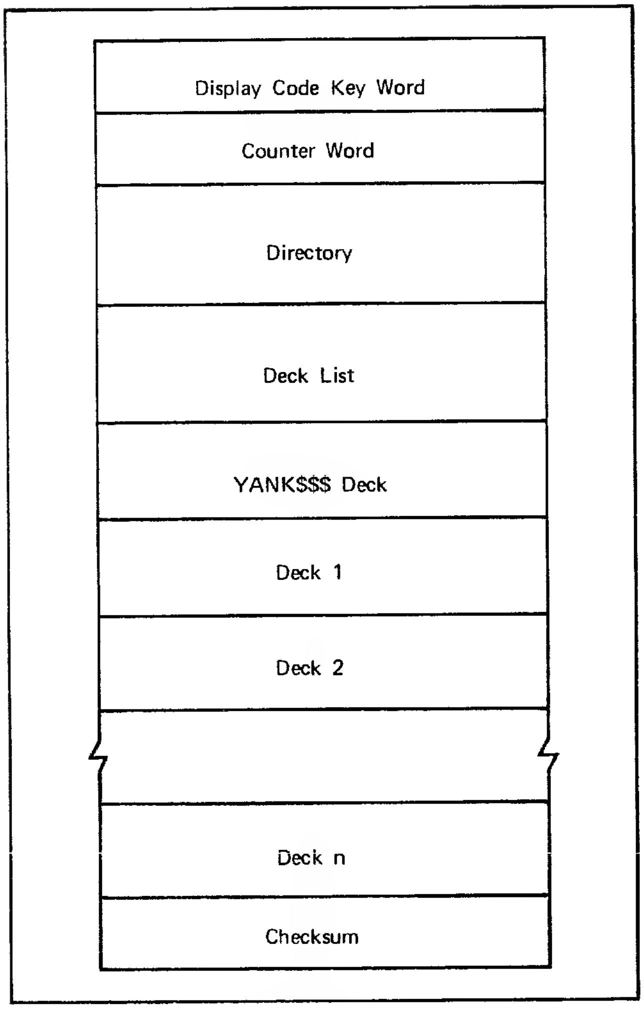


Figure D-4. Sequential Program Library Format

YANK$$$ DECK

The YANK$$$ deck is automatically created on a creation run as the first deck on the program library. It does not have a DECK card as its first card image. On correction runs, Update inserts into the YANK$$$ deck any YANK, SELYANK, YANKDECK, and DEFINE directives that it encounters during the read-input-stream phase. These directives acquire identification and sequence information from the correction set from which they originate. On a merge, the two YANK$$$ decks are merged into a single deck.

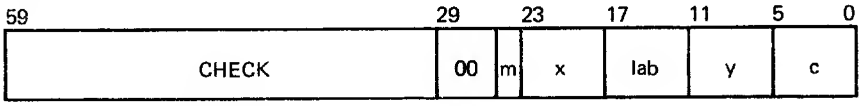
Although the YANK$$$ deck as a whole cannot be yanked or purged, cards in the deck can be deleted, yanked, or purged from it. If information other than the four directive types mentioned inadvertently gets into the YANK$$$ deck, it can be purged through the E option on the Update control statement.

DECK LIST

The deck list is a table that contains an entry for each deck on the program library. Each entry on a seqeuntial program library consists of one word containing the deck name; bit three is reserved for the deck bit that indicates whether or not the deck is a common deck. Each deck list entry on a random program library consists of two words as shown in figure D-8.

DIRECTORY

The directory is a table that contains one entry for each DECK, COMDECK, and IDENT that has ever been used for this library. Directory entries each consist of one word containing the 1 through 9 character identifier in display code, left-justified with zero-fill. Correction set identifiers and deck names are listed chronologiclaly as they are introduced into the library.



CHECK Identifies the file as being a sequential file.

m Indicates presence of deck bits in deck list:

1 deck bits present
other deck bits not present

x Character set identifier determined by IP.CSET parameter:

3 (36_8) IP.CSET is set for a 63 character set
4 (37_8) IP.CSET is set for a 64 character set

lab Label flag:

L indicates labeled tape
null indicates unlabeled tape

SCOPE 2 does not recognize tape labels.

y Indicates which character set used when the library was generated:

Y or null 64 character set used
other 63 character set used

c Indicates master control character in use when the library was created.

Figure D-5. Display Code Key Word Format

A deck name that has been purged remains in the table although it is not printed on the listable output file. The purged deck names are not removed from the table unless the E (edit) parameter is specified on the Update control statement.

The number of identifiers in the directory is limited by the amount of central memory (or small core memory) available.

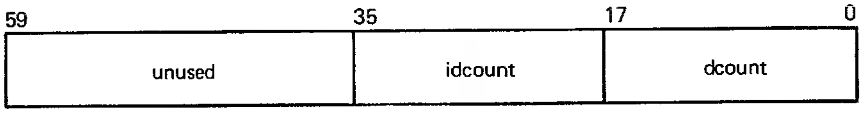
Each directory entry has the format shown in figure D-9. For a purged identifier, bits 59 through 6 are zeros, and bits 5 through 0 contain a 20_8.

COMPRESSED TEXT FORMAT

Text is an indefinite number of words that contain a correction history and the compressed image of each card in the deck. Information for each card is in the format shown in figure D-10.

OLD SEQUENTIAL FORMAT

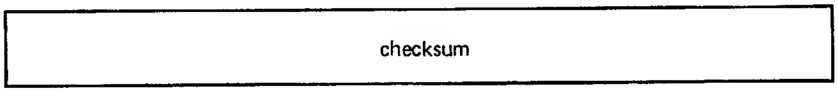
Update accepts library files in the old (pre-SCOPE 3.3) Update sequential format as shown in figure D-11. These libraries resemble the new sequential format but do not



idcount Number of identifiers in the directory.

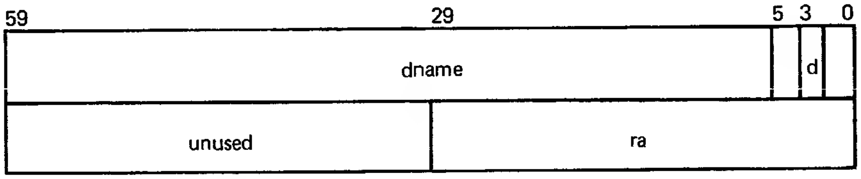
dcount Number of deck names in the deck list.

Figure D-6. Counter Word Format



checksum Count of bits in the program library.

Figure D-7. Checksum Format



dname 1 through 9 alphanumeric character deck name obtained from DECK or COMDECK directive when deck was placed on library. The first dname is YANK$$$.

d Deck bit. Indicates kind of deck.

0 common deck
1 regular deck

ra Random address of first word of compressed text for the deck.

Figure D-8. Random Program Library Deck List Format

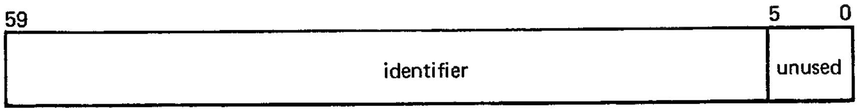
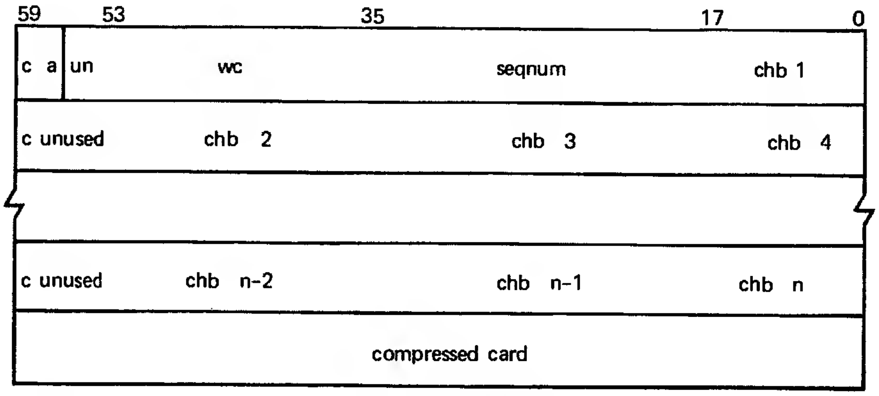


Figure D-9. Directory Format



c Correction history byte flag. Indicates the last word containing correction history bytes.

0 Not last word
1 Last word

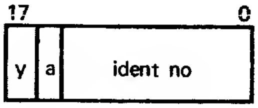
a Activity bit for the card.

0 Card is inactive
1 Card is active

wc Number of words of compressed text for this card, excluding words containing correction history bytes.

seqnum Sequence number of card (octal) according to position in deck or correction set identified by chb 1.

chb Correction history byte. Update creates a byte for each correction set that changes the status of the card. The format of chb is.



y Yank bit:

0 Card not yanked
1 Card has been yanked

a Activity bit:

0 Correction set deactivated the card
1 Correction set activated the card

identno Index to the entry in the directory that contains the name of the correction set or deck that introduced the card or changed the card status.

Compressed card The compressed image of the card in display code. Single and double spaces are unaltered. Three or more embedded spaces are replaced in the image as follows:

3 spaces replaced by 0002
4 spaces replaced by 0003
5 spaces replaced by 0004
. . .
. . .
. . .
64 spaces replaced by 0077$_8$
65 spaces replaced by 007755$_8$
66 spaces replaced by 00775555$_8$
67 spaces replaced by 00770002$_8$, and so forth

When a space is the first character of a line, it is always represented as 55$_8$ even when it is part of a string of spaces.

Trailing spaces are not considered as embedded and are not included in the card image. A four-digit octal code 0000 or word count (wc) reached marks the end of the card. This is conditional on the CHAR64 option.

When the full-character set installation option is assembled, a byte of 0001 represents a colon.

Figure D-10. Compressed Text Format on Program Library contain the CHECK word or checksun, and the text format and correction history bytes are different. Word 2 on the new format is the same as word 1 on the old format. Update no longer generates this obsolete sequential format.

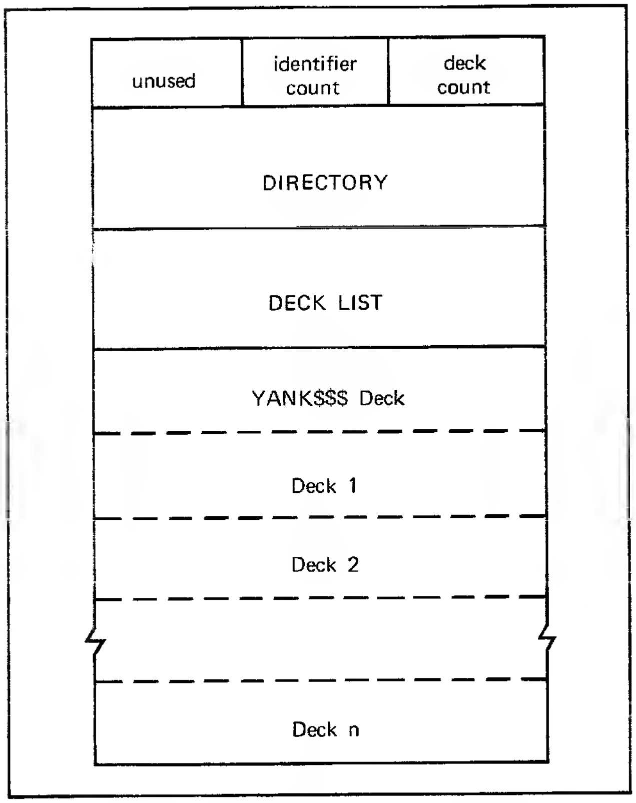


Figure D-11. Old Sequential Program Library Format

INTERCHANGEABILITY OF LIBRARIES

Random format libraries have limited interchangeability among the operating systems when they have been copied to tapes. This interchangeability is shown in table D-2.

Sequential program libraries are interchangeable among operating systems when they are system-logical-records (Record Manager type S records).

TABLE D-2. FILE INTERCHANGEABILITY

System That Generated Random Library on Tape	System to Read Random Library From Tape†		
	NOS	NOS/BE	SCOPE 2
NOS 1	Yes	No	No
NOS/BE 1	Yes	Yes	No††
SCOPE 2	No	No	Yes

†A yes indicates the tape can be read; a no indicates it cannot.

††Must be copied to unblocked mass storage file when read in.

COMPILE FILE FORMAT

Through control statement parameters, the user can specify whether the text on the compile file is to be compressed or expanded, and sequenced or unsequenced. The expanded compile file format for each card consists of 72 or 80 columns of data followed by 0 to 18 columns of sequence information. The maximum size of a card image is 90 columns.

Update attempts to place sequence information in the columns remaining in the card image after the data columns have been allocated. When the data field is 72 and the card image is 90 columns, column 73 is blank and 17 columns are available for sequencing information. In this case, the 1 to 9 character identifier is left-justified in column 74, and the sequence number is right-justified in column 86.

When the data field is 72 and the card image is 80 columns, 8 columns are available for sequencing information. If the data field is 80 and the card image is 90, 10 columns are available for sequencing information. In either of these cases, if the identifier and sequence number exceed the field, Update truncates the least significant (right most) characters of the identifier leaving the sequence number intact.

If the data field and card image are both 80, the compile file output cannot have sequence information appended.

The example in figure D-12 shows how Update positions sequencing information for the various control statement options.

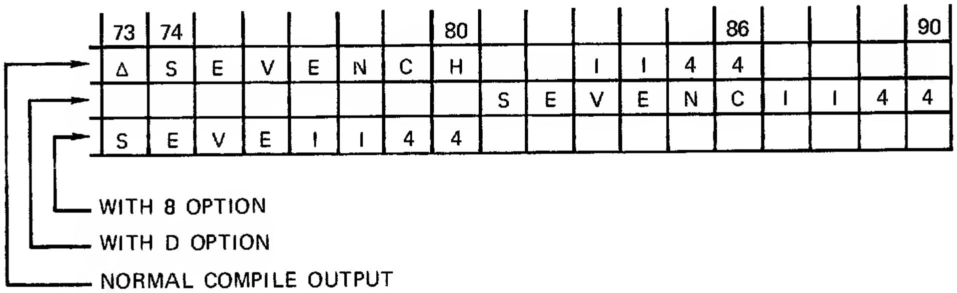
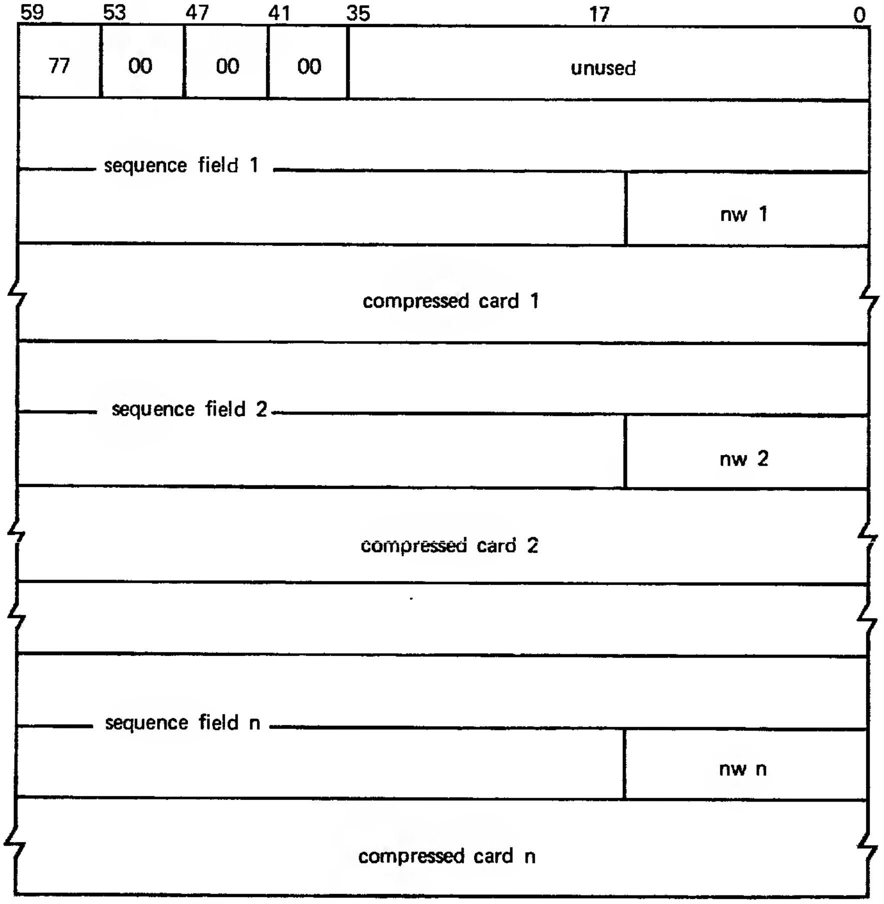


Figure D-12. Sequencing Format for Compile File

If the 80 (90) character card image on the compile file has two blanks as the last two characters, these are converted to a 0000 line terminator and the card image is 8 (or 9) words long. If the last two columns do not contain blanks, a word containing 8 blanks and a zero byte line terminator are added, thus making the card image 9 (or 10) words long. This same procedure is used for creation of the soruce file.

The format of the compressed compile file is shown in figure D-13. The first word is a loader prefix table (77_8). Compressed format is generated through the X option on the UPDATE control statement.



sequence field — 17 characters comprising card columns 74 through 90. Column 73 is always blank.

nw — Binary number of words in compressed card$_i$.

compressed card — Columns 1 through 72 of a COMPASS source card in compressed form. That is, each 00 character is replaced by the 12-bit value 0001, and three or more consecutive blanks (to a maximum of 64) are replaced by a 12-bit value 0002 through 0077_8. A single blank is represented in display code (55_8); two consecutive blanks are represented by the 12-bit value 5555_8. If the last word is not full, it is padded on the right with binary zeros. Because word count nw is present, an extra all-zero word is not required to guarantee 12 zero bits.

Figure D-13. Compile File Compressed Format

INSTALLATION OPTIONS E

The following Update features are available through assembly options.

DECLKEY Enables DECLARE directive (section 3).

CHAR64 SUPPORTS full 64-character set (refer to compressed text description appendix D).

PMODKEY Enables PULLMOD directives and G option (sections 3 and 4).

AUDITKEY Allows audit functions (section 4).

EDITKEY Allows merge and edit (section 4).

OLDPLKEY Enables Update to read both old-style and new-style old program libraries (appendix D).

SCOPE33 Declares that interface is with SCOPE 3.3 or later system, if SCOPE33 is defined. Otherwise, interface is with earlier versions.

EXTOVLP Enables detection of four types of overlap involving two or more cards in a correction set (appendix B).

DYNAMFL Declares dynamic table expansion. When this option is assembled, Update automatically expands tables as required and dynamically requests the operating system to change the user field length to accommodate the additional table area. At the end of the run, the field length is reduced to that requested by the user.

An attempt to use features when the option has not been assembled causes Update to issue error messages. For example, when PMODKEY is not set, the PULLMOD directive is not recognized as a legal directive. Refer to the installation Handbook for more details.

INDEX

CUT ON THIS LINE

COMMENT SHEET

CONTROL DATA CORPORATION

TITLE: UPDATE 1 Reference Manual

PUBLICATION NO. 60449900 REVISION B

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

General comments:

FROM NAME: ______________________ POSITION: ______________________

COMPANY NAME: ______________________

ADDRESS: ______________________

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO P,OSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD FOLD

STAPLE STAPLE

Manual Title UPDATE 1 Reference Manual ______ Pub. No. 60449900 ______ Rev. B

As part of Control Data's continuing quality improvement program, we invite you to complete this questionnaire so that you may have a more direct influence on the manuals you use.

Please rate this manual for each general and individual category on a scale of 1 through 5 as follows:

1 - Excellent 2 - Good 3 - Fair 4 - Poor 5 - Unacceptable

I. Writing Quality ______

A. Technical accuracy ______
B. Completeness ______
C. Audience defined properly ______
D. Readability ______
E. Understandability ______
F. Organization ______

II. Examples ______

A. Quantity ______
B. Placement ______
C. Applicability ______
D. Quality ______
E. Instructiveness ______

III. Format ______

A. Type size ______
B. Page density ______
C. Art work ______
D. Legibility ______
E. Printing/Reproduction ______

IV. Miscellaneous ______

A. Index ______
B. Glossary ______

V. Please provide a yes or no answer regarding manuals in general:

A. I prefer that a manual on a software product be as comprehensive as possible; physical size is of little importance. ______

B. I prefer that information on a software product be covered in several small manuals, each covering a certain aspect of the product. Smaller manuals with limited subject matter are easier to work with. ______

C. I am interested primarily in reference manuals designed for ease of locating specific information. ______

D. I am interested primarily in user guides designed to teach the user about a product or certain capabilities of a product. ______

VI. We recognize that we have a wide variety of users. Please identify your primary area of interest or activity:

A. Student ______
B. Applications programmer ______
C. Systems programmer ______
D. How many years programming experience do you have? ______
E. What languages
1. Algol ______
2. Basic ______
3. Cobol ______
4. Compass ______
5. Fortran ______
6. PL/I ______
7. Other ______

F. Have you ever worked on non-CDC equipment? ______

1. If yes, approximately what percent of your experience is on non-CDC equipment? ______

2. How do you rate CDC manuals against other similar manuals using the 1-5 ratings. (Example: XYZ Corp. 2 means XYZ manuals are good as compared to CDC manuals.)
Burroughs ______
DEC ______
Hewlett-Packard ______
Honeywell ______
IBM ______
NCR ______
Univac ______
Other ______ ______

General Comments ______

______

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

FOLD FOLD

STAPLE STAPLE

UPDATE CONTROL STATEMENT PARAMETERS

UPDATE(p1,p2, . . . ,pn)

A Sequential-to-Random Copy

omitted	no copy
A	copy

B Random-to Sequential Copy

omitted	no copy
B	copy

C Compile File Name

omitted	
or C	COMPILE
C=lfn	lfn
C=PUNCH	PUNCH
C=0	none

D Data Width On Compile File

omitted	72 columns
D	80 columns

E Edit Old Program Library

omitted	no editing
E	editing

F Full Update Mode

omitted	normal selective mode
F	full mode

G Pullmod File Name

omitted	source file
G=lfn	lfn

H Character Set Change

omitted	
or H	default set
H=3	63
H=4	64

I Input Stream File Name

omitted	
or I	INPUT
I=lfn	lfn

K Compile File Sequence

omitted	C parameter determines deck location
K	COMPILE directive sequence on file COMPILE
K=lfn	COMPILE directive sequence on file lfn

L Listable Output Options

omitted	creation run: A, 1, 2
	correction run: A, 1, 2, 3, 4
	copy run: A, 1
L=0	suppress listing
L=c. . .c	options 1 thru 9, A or F

M Merge File Name

omitted	no merge
M	MERGE
M=lfn	lfn

N New Program Library Name

omitted	
or N	NEWPL
N=lfn	lfn

O Listable Output File Name

omitted	
or O	OUTPUT
O=lfn	lfn

P Old Program Library Name

omitted	
or P	OLDPL
P=lfn	lfn
P=lfn/s1/	
s2/ . . .	lfn; secondaries on si
P=/s1/s2. . .	OLDPL; secondaries on si

Q Quick Update Mode

omitted	normal selective mode
Q	quick mode

R Rewind Files

omitted	rewind files
R	no rewinding
R=c. . .c	rewind specified files (C, N, P, S)

S Source File Name

omitted	none
S	SOURCE
S=lfn	lfn

T Omit Common Decks From Source File

omitted	none
T	SOURCE
T=lfn	lfn

U Debug Help

omitted	fatal error ends execution
U	fatal errors do not end execution

W Sequential New Program Library Format

omitted	random if possible
W	sequential

X Compressed Compile File

omitted	not in compressed format
X	in compressed format

8 Card Image Width On Compile File

omitted	90 columns
8	80 columns

* Master Control Character

omitted	*
*=c	c

/ Comment Control Character

omitted	/
/=c	c

CORPORATE HEADQUARTERS P.O. BOX 0 MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION